

BRUCKMANN-LOTHAR-ENGLISH-GERITS

MICRO APPLICATION

6

AMSTRAD

**LA BIBLE
DU PROGRAMMEUR
DE L'AMSTRAD CPC**



UN LIVRE DATA BECKER

BRUCKMANN-LOTHAR-ENGLISH-GERITS

MICRO APPLICATION

6

AMSTRAD

**LA BIBLE
DU PROGRAMMEUR
DE L'AMSTRAD CPC**



UN LIVRE DATA BECKER

Distribué par MICRO APPLICATION
147 Av. Paul Doumer
92500 RUEIL-MALMAISON

et également

EDITIONS RADIO
3 rue de l'Eperon
75006 PARIS

(c) Reproduction interdite sans l'autorisation de MICRO APPLICATION.

"Toute représentation ou reproduction, intégrale ou partielle, faite sans le consentement de MICRO APPLICATION est illicite (loi du 11 mars 1957, alinéa 1er de l'article 40).

Cette représentation ou reproduction illicite, par quelques procédés que ce soit, constituerait une contrefaçon sanctionnée par les articles 425 et suivants du Code Pénal.

La loi du 11 mars 1957 n'autorise, aux termes des alinéas 2 et 3 de l'article 41, que les copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à l'utilisation collective d'une part, et d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration".

ISBN 2-86899-011-8

Copyright (c) 1985 DATA BECKER
Merowingerstr. 30
4000 Düsseldorf
Allemagne de l'Ouest

Copyright (c) Traduction française 1985 MICRO APPLICATION
147 av. Paul Doumer
92500 RUEIL MALMAISON

Traduction Française et mise en pages assurées par Pascal HAUSMANN

Edité par Frédérique BEAUDONNET
Léo BRITAN
Philippe OLIVIER

TABLE DES MATIERES

	Introduction.....	1
1	ELECTRONIQUE.....	3
1.1	Ce que vous devez savoir sur votre machine.....	3
1.1.1	L'organisation de la mémoire.....	4
1.1.2	Les RSTs.....	7
1.2	Le processeur.....	10
1.2.1	Les connexions du Z80.....	11
1.2.2	Description des registres du Z80.....	15
1.2.3	particularités du Z80 du CPC.....	19
1.3	Le Gate Array.....	23
1.3.1	Les connexions du GA.....	25
1.3.2	Description des registres du GA.....	29
1.4	Le contrôleur vidéo.....	33
1.4.1	Pinout du CRTC.....	35
1.4.2	Description des registres du CRTC.....	36
1.5	La Ram sur le CPC.....	40
1.6	Ram vidéo entre Z80 et 6845.....	46
1.7	8255.....	51
1.7.1	Les connexions du 8255.....	51
1.7.2	Les modes de travail du 8255.....	52
1.7.3	Description des registres du 8255.....	53
1.7.4	Particularités du 8255 sur le CPC.....	55
1.8	Le chip sonore.....	60
1.8.1	Les connexions du 8912.....	61
1.8.2	Description des registres du 8912.....	63
1.8.3	Particularités du 8912 sur le CPC.....	66
1.9	Les interfaces.....	70
1.9.1	Le clavier.....	70

1.9.2	La connexion du moniteur.....	71
1.9.3	Le lecteur de cassette.....	73
1.9.4	Le port imprimante.....	78
1.9.5	Le port joystick.....	81
1.9.6	Le connecteur d'extension.....	82
2	SYSTEME D'EXPLOITATION.....	85
2.1	Les vecteurs du système d'exploitation.....	86
2.2	La Ram du système d'exploitation.....	95
2.3	Utilisation de routines avec l'exemple du hardcopy.....	99
2.4	Le traitement des interruptions dans le système d'exploitation.....	111
2.5	Le listing du système d'exploitation.....	115
2.5.1	Kernel.....	115
2.5.2	Machine Pack.....	143
2.5.3	Jump Restore.....	154
2.5.4	Screen Pack.....	162
2.5.5	Text screen.....	188
2.5.6	Graphics screen.....	212
2.5.7	Keyboard Manager.....	229
2.5.8	Sound Manager.....	II 1
2.5.9	Cassette Manager.....	II 22
2.5.10	Screen Editor.....	II 50
2.6	Le générateur de caractères.....	II 67
3	BASIC.....	II 90
3.1	L'interpréteur.....	II 90
3.2	La pile Basic.....	II 96
3.3	Les vecteurs Basic.....	II 100
3.4	La Ram Basic.....	II 104
3.5	Basic et langage-machine.....	II 107
3.5.1	L'instruction CALL.....	II 107
3.5.2	Extensions RSX.....	II 108
3.5.3	Le pointeur de variable.....	II 111
3.6	Le listing de la Rom Basic.....	II 114
3.6.1	L'arithmétique avec virgule flottante.....	II 118
3.6.2	L'arithmétique avec nombres entiers.....	II 144
3.6.3	L'interpréteur Basic.....	III 1

4	ANNEXES.....	IV	1
4.1	Les routines du système d'exploitation.....		1
4.2	Références à la Ram du système.....		11
4.3	Les routines de la Rom Basic.....		16
4.4	Les tokens Basic.....		25

Schémas

INTRODUCTION

Lorsque nous avons reçu le premier CPC à l'automne 1984, nous avons été d'abord assez sceptiques. 'Un parmi tant d'autres' avons-nous pensé avant de découvrir la puissance de cet ordinateur.

La taille du présent ouvrage ainsi que son contenu montrent que nous avons vite changé radicalement de point de vue.

Le CPC est une machine fantastique qui offre actuellement un rapport entre le prix et les possibilités de l'ordinateur qui n'a pas de concurrent. Dans la classe de prix qui est la sienne, le CPC représente une nouvelle dimension. Plusieurs points sont décisifs à cet égard: d'abord, le fait qu'il s'agisse d'un système complet. Grâce au moniteur livré avec l'appareil, pas de dispute pour savoir si on regarde Dallas ou si on utilise l'ordinateur. De même, le lecteur de cassette intégré rend inutiles les câbles de connexion, le réglage du volume et les interfaces qui faisaient de l'utilisation du lecteur de cassette un problème permanent. Votre ordinateur possède tout ce dont vous avez besoin pour pouvoir l'utiliser immédiatement.

Les possibilités de l'ordinateur sont un second point fort de ce matériel. Le Basic LOCOMOTIVE est certainement le meilleur disponible sur les ordinateurs de cette catégorie. La programmation des interruptions très souple et très facile d'emploi dont dispose ce Basic est certainement un des aspects les plus remarquables de cet ordinateur. L'excellence du graphisme et la possibilité d'avoir un écran en 80 colonnes sans module ni coût supplémentaire est pour l'heure sans équivalent, alors que d'autres ordinateurs de la même catégorie ont déjà du mal à présenter sur l'écran 40 caractères par ligne parfaitement lisibles.

La résolution graphique de 640 points sur 200 est également unique pour cette catégorie de prix. On ne trouve de possibilités comparables que sur l'IBM PC qui est tout de même au moins cinq fois plus cher que le CPC. Les possibilités sonores du CPC sont également impressionnantes.

En ce qui concerne la vitesse, le CPC n'a pas à rougir. Le microprocesseur intégré Z80 fonctionne avec une fréquence de 4 mégahertz et il dispose d'un jeu d'instructions très puissant. Ce jeu d'instructions a été exploité au maximum par les développeurs de la

machine qui ont ainsi réussi à réaliser un interpréteur Basic particulièrement rapide.

Mais les possesseurs d'un nouvel ordinateur cherchent en général très vite à obtenir plus d'informations sur leur machine. Le manuel d'utilisation du CPC, qui est par ailleurs tout à fait remarquable, ne suffit pas à répondre à l'attente de ceux qui veulent connaître leur ordinateur dans les moindres détails et notamment pour ceux, pour qui le Basic a perdu un peu de son attrait, qui en ont découvert les limites et qui souhaiteraient donc s'attaquer à la programmation en langage-machine. Il faut alors disposer d'informations qui dépassent largement le cadre du manuel d'utilisation.

Ce sont ces informations que le présent ouvrage met à votre disposition. Ce livre est le résultat de nuits et de journées de travail consacrées au CPC.

Vous trouverez ici une description détaillée du matériel (hardware) avec un schéma, un listing du système d'exploitation et du Basic très complètement commenté, les adresses importantes de la RAM mais aussi des instructions Basic qui ne sont pas décrites dans le manuel. Vous trouverez également de petits trucs concernant l'utilisation du lecteur de cassette et de l'imprimante ainsi que la programmation du graphisme en langage-machine.

Nous espérons que les informations que nous vous fournissons vous seront utiles et vous permettront de connaître encore mieux votre CPC.

Les auteurs

1 LE MATERIEL (HARDWARE)

1.1 Ce que vous devez absolument savoir sur votre machine

Vous n'avez pas encore pris votre tournevis pour observer la vie interne de cette "boîte magique"? Cela ne fait rien, nous vous avons évité ce travail de dévissage et nous avons photographié le résultat. L'illustration 1.1.0.1 montre à quoi ressemble l'intérieur de votre machine.

Ce ne sont pas plus de 25 circuits intégrés qui sont disposés sur une plaque de taille importante. Ce n'est donc pas à une électronique particulièrement coûteuse que le CPC doit sa puissance et c'est plutôt la partie logiciel (software) qui rend cet ordinateur extraordinaire et qui explique également le prix particulièrement bas auquel le système complet est proposé. Les quelques composants électroniques qui constituent le CPC ne reviennent en effet pas très cher.

Seuls 9 circuits intégrés représentent la mémoire dont dispose votre CPC. Huit composants du type 4164 constituent les Rams, la mémoire de travail de l'ordinateur. Le neuvième circuit intégré de mémoire, est une ROM de 32 kilo-octets. Le processeur Z80 du CPC ne peut cependant, comme tout processeur 8 bits adresser qu'une zone de 64 kilo-octets et cette zone est entièrement remplie par les composants Ram.

L'adressage apparemment impossible de 96 kilo-octets a cependant été obtenu grâce à un truc de programmation très subtile connu sous le nom de bank-switching (commutation de banques de mémoire). Mais ce n'est pas tout! Théoriquement, il est possible de connecter au CPC jusqu'à un maximum de 252 ROMs externes de 16 K chacune, qui pourraient alors être adressées par bank-switching. La zone ainsi adressable est donc d'environ 4 méga-octets!

Le CPC contient en outre comme composants hautement intégrés un video-controller HD 6845, un port parallèle 8255, un chip sonore AY-3-8912 et un élément appelé gate array, qui a été développé spécialement pour le CPC.

Le contrôleur vidéo a pour fonction de fournir tous les signaux nécessaires pour le fonctionnement du moniteur. Il adresse également la

mémoire-écran, cette zone de la mémoire dans laquelle sont placés les caractères à représenter et le graphisme. Il produit également le refresh qui est nécessaire pour les Rams, sans lequel vous perdriez vite les informations stockées.

La tâche du chip sonore est définie par le nom de ce composant. Le choix des constructeurs est très bon. Le AY-3-8912 a été utilisé dans de nombreux ordinateurs parce qu'il est très polyvalent et qu'il permet des possibilités étendues d'influencer le son.

Le 8255 est le "travailleur de force" du CPC. Ses tâches sont très diverses.

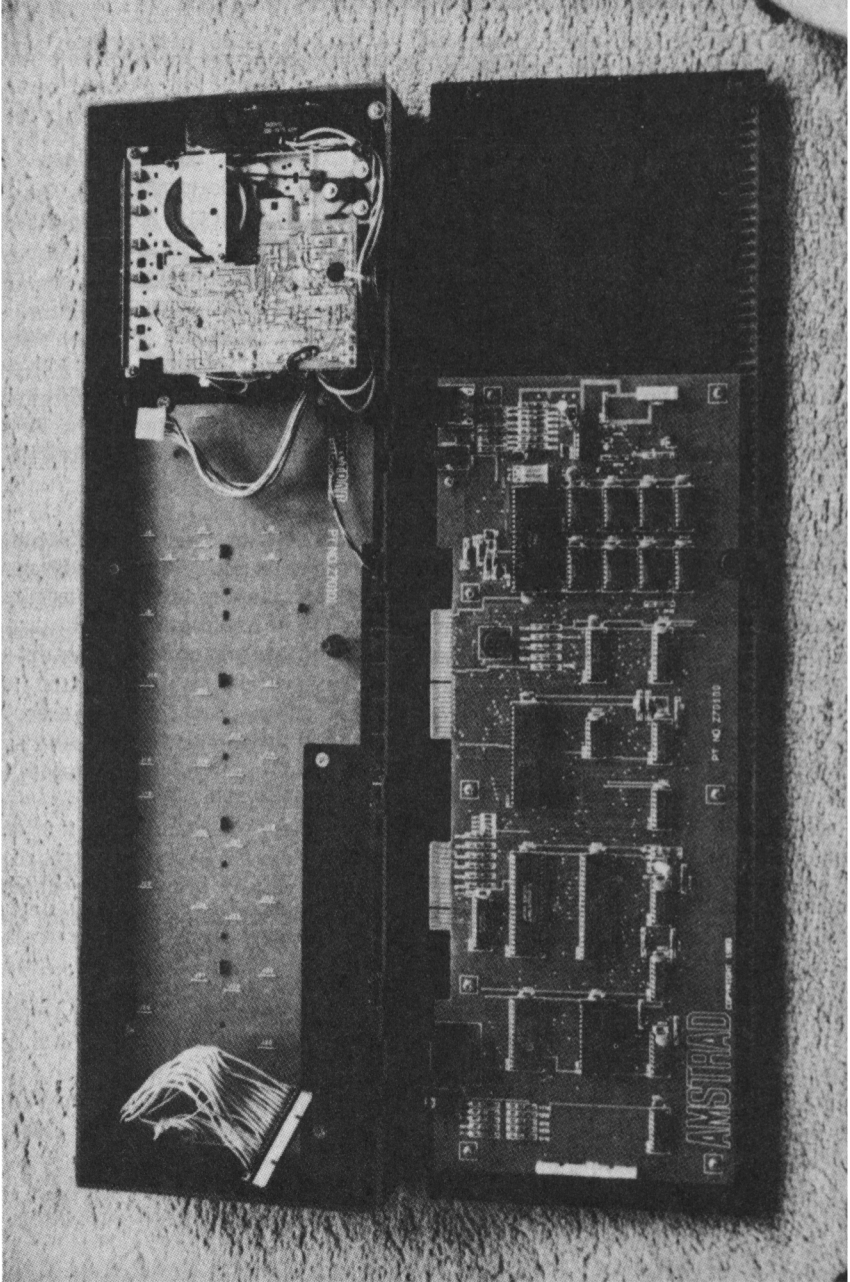
Cela va du contrôle du clavier à la commande du chip sonore en passant par la commande du magnétophone, à la détermination de certaines possibilités du CPC etc...

Le gate array est particulièrement intéressant. Ce composant commande tant de choses dans le CPC qu'on pourrait presque le qualifier de processeur auxiliaire. C'est ainsi qu'il prend en charge bon nombre des tâches concernant l'écran, telles que la représentation des différentes couleurs et les différents formats de l'écran. Tous les signaux nécessaires de synchronisation sont produits par le gate array. Les interruptions, qui interrompent le déroulement normal des programmes 300 fois par seconde, sont produites par le gate array ainsi que les signaux nécessaires à la gestion de la mémoire 96 K du CPC.

Le schéma 1.1.0.2 montre comment les différents composants travaillent ensemble.

1.1.1 La disposition de la mémoire

Il y a encore 5 ans, les ordinateurs disposant de 16 K de RAM étaient considérés comme bien armés. Mais depuis l'apparition du Commodore 64, les limites de la mémoire ont été nettement repoussées. Un constructeur de micro-ordinateurs n'a de chances suffisantes de prendre une part du marché que si les magiques 64 apparaissent sur sa machine.



Le CPC dispose lui aussi d'une RAM de 64 K = 65536 cases mémoire. Il possède en plus une ROM intégrée de 32 K.

D'ailleurs, il n'est pas très difficile de placer une mémoire de 64 K dans un ordinateur puisque les processeurs 8-bits, qui sont les plus répandus, peuvent tous adresser une zone de 64 kilo-octets. Le Z80 du CPC peut lui aussi adresser 64 K de mémoire sans truc particulier. Mais cela suffit normalement tout juste pour la mémoire RAM et c'est tout.

Il faut donc recourir à un procédé spécial, le bank-switching, si l'on veut pouvoir adresser plusieurs mémoires avec ce type de processeurs. Ce procédé permet en effet de choisir entre des zones de mémoire (qu'on appelle banques) ROM et RAM qui se chevauchent. Il s'agit d'un procédé qui n'utilise pas de solution matériel mais a uniquement recours à un logiciel qui organise la cohabitation de la ROM et de la RAM aux mêmes adresses. Cette solution logiciel a été remarquablement mise en oeuvre par les développeurs de l'ordinateur.

Le CPC présente donc l'image suivante: 64 K de RAM sont adressés directement. Parallèlement à la RAM se trouvent une moitié de la ROM dans les 16 K inférieurs (&0000 à &3FFF) et l'autre moitié de la ROM dans les 16 K supérieurs (&C000 à &FFFF).

Les 16 K inférieurs de ROM contiennent le système d'exploitation et un bloc de routines arithmétiques. Dans le système d'exploitation se trouvent toutes les routines dont le CPC a besoin pour lire par exemple un caractère tapé au clavier, pour placer un caractère ou un point sur l'écran mais c'est également le système d'exploitation qui commande le lecteur de cassette et l'interface imprimante ainsi que le son.

Dans les 16 K supérieurs se trouve l'interpréteur Basic. Ces 16 K n'ont pas de fonction spéciale. Il est possible de connecter dans cette zone jusqu'à 252 ROMs supplémentaires. C'est ainsi que les routines nécessaires pour la gestion du lecteur de disquette sont placées dans une ROM qui 'partage' cette zone avec le Basic.

La disposition de la mémoire est représentée par la figure 1.1.1.1

1.1.2 Extension d'Instructions à travers RST

Etant donné ce mode de gestion de la mémoire, on peut cependant se demander comment peut se faire l'accès aux ROMs ou aux RAMs situées dans les mêmes zones. Pour éviter aux utilisateurs le travail de programmation assez considérable que nécessiteraient normalement ces tâches, les programmeurs du système d'exploitation ont eu une riche idée. Grâce à des programmes spéciaux et à une utilisation habile des instructions RESTART du Z80, ils ont pratiquement abouti à faire des restarts RST1 à RST5 une extension du jeu d'instructions du Z80. Ces RSTs peuvent être employés comme des JPs ou des CALLs ordinaires. Certains RSTs réclament par ailleurs une adresse sur 3 octets. Le troisième octet, supplémentaire, détermine dans quelle ROM le JP ou le CALL doit aller.

LOW JUMP RST 1

Cette instruction Restart permet d'appeler une routine du système d'exploitation ou de la RAM située dans la même zone d'adresses. L'instruction RST doit être suivie immédiatement par l'adresse de la routine à appeler. Comme 14 bits suffisent pour définir une adresse comprise entre 0 et &3FFF, les deux bits supérieurs restants sont utilisés pour sélectionner la ROM ou la Ram:

Bit 14=0 Sélection du système d'exploitation
Bit 14=1 Sélection de la Ram
Bit 15=0 Sélection de la ROM Basic
Bit 15=1 Sélection de la Ram

Un appel de la routine système pourrait donc se présenter ainsi:

```
RST 1  
DW &1410+&8000
```

Le bit 15 mis sélectionne la RAM dans la zone de &C000 à &FFFF, alors que le bit 14 annulé appelle le système d'exploitation.

Le code à l'adresse 8 est constitué uniquement par un saut à l'adresse &B982.

SIDE CALL RST 2

Cette instruction Restart permet d'appeler une routine d'une ROM d'extension. Cette instruction est utilisée lorsqu'un programme sous forme d'un module de ROM nécessite plus de 16 kilo-octets et ne peut pas tenir dans un seul module d'extension. Le SIDE CALL permet alors d'appeler une routine se trouvant dans la seconde, la troisième ou la quatrième ROM appartenant au programme, sans qu'il soit pour cela nécessaire de connaître le numéro absolu de la ROM qu'il s'agit d'appeler ainsi. L'instruction RST 2 doit être suivie de l'adresse de la routine - &C000, c'est-à-dire de l'adresse relative par rapport au début de la ROM. Les deux bits supérieurs servent à sélectionner l'une des quatre ROMs différentes utilisées.

Le code à l'adresse &0010 est constitué uniquement par un saut à l'adresse &BA16.

FAR CALL RST 3

Cette instruction Restart permet d'appeler une routine n'importe où en ROM ou en RAM. L'instruction RST 3 doit être suivie de l'adresse sur deux octets d'un bloc de paramètres composé de trois octets. Les deux premiers de ces octets-paramètres contiennent l'adresse de la routine qui doit être appelée et le troisième octet doit contenir l'état ROM/RAM souhaité. Les valeurs de 0 à 251 permettent d'appeler une ROM supplémentaire et les quatre valeurs restantes ont la fonction suivante:

Valeur	&0000-&3FFF	&C000-&FFFF
252	Système d'exploitation	Basic
253	Système d'exploitation	RAM
254	RAM	Basic
255	RAM	RAM

Le code à l'adresse &0018 est constitué uniquement par un saut à l'adresse &B9BF.

1.2 Le processeur Z80

Le début des années 70 a connu le triomphe des microprocesseurs. La société INTEL a pu se tailler avec le processeur 8080 une part significative du marché parce qu'au moment où elle le lança sur le marché, il n'avait pratiquement pas de concurrent dans cette catégorie. C'est bien ce qui frappe quand on examine de plus près les données techniques de ce processeur. Le 8080 avait en effet besoin de trois tensions de courant différentes et de deux circuits intégrés supplémentaires pour la production des signaux de commande et de synchronisation.

La société ZILOG a développé le Z80 dans les années 74/75. Mais au lieu de développer un processeur radicalement nouveau, on s'en tint à la conception du 8080 qui avait rencontré un tel succès. C'est pourquoi le Z80 est compatible avec le 8080 (mais non pas l'inverse). C'est-à-dire que tous les programmes écrits pour un 8080 tournent aussi sur un Z80.

Cependant toutes les particularités considérées comme néfastes du 8080 furent éliminées et le jeu d'instructions fut largement étendu. Le Z80 ne nécessite d'autre part qu'une tension de +5Volt et il n'a pas besoin de circuits intégrés externes pour produire les signaux de commande.

Mais examinons en style télégraphique les données techniques de ce processeur, avant que nous n'entrions plus dans le détail de ses caractéristiques:

- Processeur 8-bits de technologie NMOS
- Bus d'adresses 16-bits
- Alimentation unique 5 Volt
- Horloge simple
- Compatible TTL
- Fréquence d'horloge de 2.5, 4, 6 ou même 8 MHz
- Compatibilité logiciel avec le 8080
- Double jeu de registres plus deux registres d'index
- Entrée d'interruptions non-masquable
- Entrée d'interruptions masquable avec trois modes de travail
- Refresh automatique de RAMs dynamiques
- Circuits intégrés périphériques du 8080 directement connectables

Ces données techniques ainsi qu'un grande masse de logiciels disponibles ont fait du Z80 l'un des processeurs 8-bits les plus répandus. Dans le domaine des ordinateurs familiaux et personnels, seul le 6502 a obtenu une diffusion comparable.

1.2.1 Les connexions du Z80

Après ce bref aperçu sur les possibilités du Z80, intéressons-nous maintenant à l'affectation des 40 pins de connexion du Z80.

Les points de connexion du Z80 peuvent être répartis entre les 4 groupes bus de données, bus d'adresses, bus de commande et canaux de transmission.

Bus d'adresses

A0 - A15 : Lignes d'adresses; ces connexions permettent d'appeler une case mémoire dans la zone adressable qui comprend 65536 cases mémoire. Dans le traitement des instructions d'entrée-sortie, les 8 bits inférieurs de l'adresse sont utilisés pour sortir l'adresse d'entrée-sortie correspondante. 256 ports différents sont ainsi possibles. Avec certaines limites tenant au jeu d'instructions, ce sont même 65536 ports qui peuvent être adressés. Les 16 canaux d'adresse sont alors utilisés pour constituer l'adresse du port. Nous reviendrons plus tard sur ce cas particulier.

Bus de données

D0 - D7 : Lignes de données; ces canaux bidirectionnels transmettent les données venant du processeur ou allant vers le processeur. Elles font le lien entre le processeur et la case mémoire ou l'adresse de port choisies à travers le bus d'adresses.

Bus de commande

- M1* : Machine Cycle One, ce signal de commande indique que le processeur lit le code d'instruction sur le bus de données. L'étoile signifie par ailleurs pour ce signal et pour les signaux suivants, qu'il s'agit d'un signal actif avec low.
- MREQ* : Memory REQuest*, ce signal de sortie indique par un low que le processeur entreprend un accès en lecture ou écriture à une adresse de la mémoire et que l'adresse sur le bus d'adresses est valable.
- IORQ* : Input/Output ReQuest*, ce signal de sortie indique par un low que le processeur entreprend un accès en lecture ou écriture à une adresse de port et que l'adresse de port sur le bus d'adresses est valable.
- RD* : Read*, ce signal de sortie indique par un low que le processeur veut lire des données dans une case mémoire ou dans une adresse de port. L'utilisation conjointe avec MREQ* ou IORQ permet de distinguer entre la lecture de la mémoire ou d'un port.
- WR* : Write*, ce signal indique, lors d'accès en écriture du processeur à la mémoire ou aux adresses de port, que les données figurant sur le bus de données sont valables. Ici aussi, l'utilisation conjointe de WR* avec MREQ* ou IORQ* permet de distinguer si les données doivent être écrites dans la mémoire ou dans une adresse de port.
- RESET* : Lorsque ce signal d'entrée passe à low, le compteur de programme reçoit la valeur &8000, les interruptions sont interdites et le mode d'interruption 0 est activé. Dès que ce signal d'entrée redevient high, le processeur commence l'exécution du programme à partir de l'adresse &0000.
- NMI* : Non Maskable Interrupt*, ce signal d'entrée provoque toujours par un double signal high-low une interruption du programme exécuté par le processeur. Les valeurs placées en &0066 et &0067 sont alors chargées dans le compteur de programme et le programme se poursuit à partir de cet endroit.

IRQ* : Interrupt ReQuest*, ce signal d'entrée peut provoquer par un low une interruption du programme exécuté par le processeur, à condition que ce type d'interruptions soit autorisé par instruction. Les effets dépendent du type d'interruption et seront évoqués plus tard. IRQ* est, au contraire de NMI*, un signal statique qui doit persister jusqu'à ce que la demande d'interruption ait été prise en compte.

WAIT* : Ce signal permet d'adapter l'accès en lecture ou en écriture du Z80 à des mémoires plus lentes ou à des conditions spéciales du système.

BUSRQ* : BUSReQuest*; lorsque ce signal d'entrée passe à low, les canaux de données et d'adresses ainsi que tous les canaux de commande de sortie deviendront high après le traitement de l'instruction actuelle et le signal BUSAK* deviendra low. Maintenant, un second processeur pourrait prendre en charge l'accès à la mémoire et aux éléments périphériques; ce signal est cependant essentiellement utilisé pour le DMA (DMA= Direct Memory Access, transfert de données très rapide en contournant le processeur).

BUSAK* : BUSAKnowledge*, est le signal de sortie correspondant à BUSRQ*. Un low indique au DMA controller ou au second processeur que tous les signaux de commande et de bus sont high et qu'un accès est maintenant possible.

HALT* : Ce signal de sortie devient low après que le processeur ait exécuté l'instruction en langage-machine HALT. Après cette instruction, le processeur ne fait plus rien d'autres que d'exécuter des NOPs pour assurer le Refresh. Seule une interruption peut à nouveau le "réveiller".

RFSSH* : ReFrESH*, ce signal de sortie indique que les sept canaux d'adresses inférieurs contiennent une adresse de Refresh valable. Comme le processeur n'a besoin du bus d'adresses et de données qu'à certains moments, le bus d'adresses peut être utilisé le reste du temps pour rafraîchir les RAMs dynamiques, sans qu'une électronique coûteuse ou des routines spéciales de rafraîchissement ne soient pour cela nécessaires.

Horloge et alimentation électrique

0 _____ : Le signal d'entrée phi sert d'horloge pour le processeur. Comme le Z80 est un circuit intégré statique, la fréquence d'horloge peut varier entre 0 Hertz et la fréquence maximale indiquée. La forme du signal d'horloge doit cependant répondre à certaines exigences. La durée low de ce signal ne doit pas dépasser 2 microsecondes. Cette valeur n'a d'ailleurs qu'un intérêt théorique, puisqu'on essaiera évidemment toujours de fournir au processeur une fréquence d'horloge la plus élevée possible, de façon à obtenir une exécution rapide du programme.

GND _____ : Branchement à la masse du processeur.

Vcc _____ : C'est par cette connexion que le Z80 reçoit son alimentation en courant électrique continu de 5 Volts et environ 150 à 200 milliampères.

1.2.2 LA STRUCTURE DES REGISTRES DU Z80

Comme nous l'avons indiqué au début, le Z80 a été construit de telle façon que les programmes du 8080 puissent être repris sans problème. Mais le Z80 dispose d'un nombre de registres nettement supérieur.

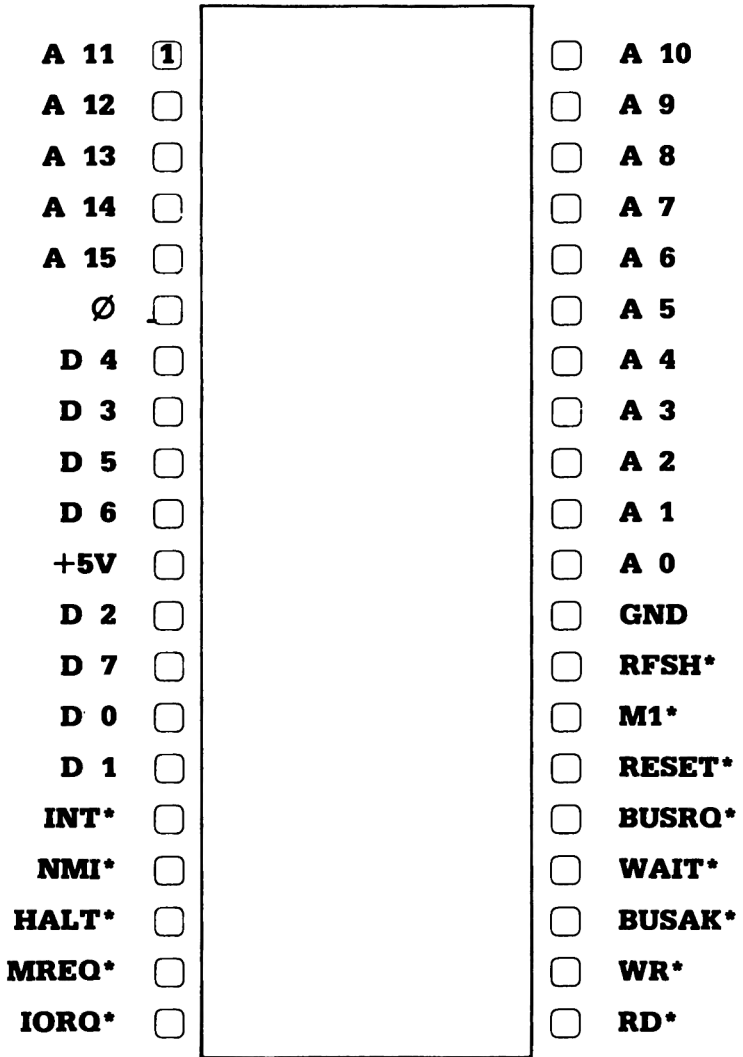
Mais qu'est-ce donc qu'un registre?

Un registre n'est rien d'autre qu'une mémoire de lecture/écriture sur le chip du processeur. Chaque processeur doit disposer d'un minimum de registres. Dans ces cases de mémoire, les données peuvent être placées, ainsi que les résultats d'instructions arithmétiques et logiques. D'autres registres ont des fonctions spéciales, telles que la gestion de la pile, ou sont utilisés comme compteur de programme.

Comme les opérations telles qu'un transfert de données entre deux registres ou l'addition des contenus de deux registres ne peuvent se faire à travers le bus de données, de telles opérations peuvent être exécutées beaucoup plus rapidement que lorsque les valeurs nécessaires doivent être recherchées dans des cases de mémoire externes.

On peut donc dire en règle générale que les processeurs disposant d'une mémoire interne plus importante sont supérieurs aux processeurs disposant de peu de registres pour le traitement des mêmes programmes car le transfert de données est toujours plus rapide à l'intérieur du processeur qu'entre le processeur et les cases de mémoire externes.

Le Z80 dispose de 22 registres au total, 18 registres de 8 bits et 4 registres de 16 bits. La figure 1.2.2.1 montre la disposition de ces registres.



1.2.1.1. Pinout du Z80

Dans cette figure, certains registres sont marqués par un cadre plus épais. Ces registres existent également sur le 8080.

Vous voyez également que la plupart des registres 8 bits apparaissent en double exemplaire. Il s'agit des registres A, F, B, C, D, E, H et L. Le programmeur peut choisir entre ces deux jeux de registres.

Nous ne parlerons à l'avenir que d'un seul jeu de registres, d'autant que le programmeur du CPC ne dispose en fait, à moins de recourir à certaines astuces particulières, que d'un seul jeu de registres. Le jeu de registres alternatif est utilisé par le système d'exploitation pour la gestion des interruptions. Mais notez bien que toutes les tâches d'un jeu de registres peuvent également être prises en charge par le jeu de registres alternatif, si celui-ci n'est pas employé pour des opérations spécifiques.

Les registres B à L sont les registres 8 bits normalement disponibles, alors que les registres A et F répondent à des tâches particulières.

Le registre A est généralement qualifié d'accumulateur. C'est dans l'accumulateur qu'on obtient le résultat de toutes les opérations arithmétiques et logiques sur 8 bits. Pour ces opérations, un opérande doit d'autre part être placé dans l'accumulateur. Pour additionner par exemple deux octets, il faut placer un opérande dans l'accumulateur alors que le second opérande peut être placé dans un autre registre du processeur ou dans une case de la mémoire externe. Après l'addition, le résultat se trouve dans l'accumulateur.

Comme, lors de telles opérations, le résultat peut être supérieur à la valeur maximale qui peut être exprimée avec 8 bits ($255+255=510$), un bit supplémentaire est nécessaire pour représenter le résultat correctement. C'est le registre F qui remplit cette fonction. Le registre F, généralement qualifié de registre flag est divisé en ses différents bits. Un de ces bits a entre autre pour fonction de conserver une éventuelle retenue (carry en anglais) résultant de telles additions. Les autres bits indiquent si le résultat d'opérations de calcul ou de comparaisons est nul, etc...

Les registres B à L ne peuvent toutefois pas uniquement être appelés séparément. B et C, D et E ainsi que H et L peuvent être regroupés en registres 16 bits. Ces registres 16 bits reçoivent alors naturellement les noms BC, DE et HL. Les registres doubles conviennent parfaitement à

l'adressage de tableaux ainsi qu'au transfert et à la recherche de blocs de données.

Le registre double HL a une signification particulière. Comme le Z80 dispose d'instructions d'addition et de soustraction sur 16 bits, le registre HL fait office, pour de telles instructions, d'accumulateur 16 bits.

Les registres PC, SP, IX et IY ne travaillent qu'avec des valeurs 16 bits (remarque: les spécialistes savent qu'il est également possible de manipuler les registres d'index octet par octet mais nous ne considérerons IX et IY que comme de purs registres 16 bits).

Le registre PC est le compteur de programme (Programm Counter). Le contenu du PC est placé sur le bus d'adresse comme adresse pour les mémoires externes. Avec chaque instruction, le PC est incrémenté (augmenté de 1) automatiquement. Pour les instructions sur plusieurs octets, le PC est automatiquement augmenté de la valeur correspondant à ce nombre d'octets. Si des sauts doivent se produire à l'intérieur d'un programme, la nouvelle adresse du programme est automatiquement chargée dans le PC et le processeur continue l'exécution à partir de cette adresse.

Le registre SP est le pointeur de pile (Stack Pointer). La pile est utilisée lorsque des sous-programmes sont appelés. Dans ce cas en effet, l'adresse de retour est automatiquement placée sur la pile puis rechargée dans le PC après exécution du sous-programme.

Les deux registres 16 bits IX et IY permettent grâce à des instructions spéciales un travail particulièrement efficace avec les tableaux.

Il ne reste plus que les registres I et R. Le registre I ou registre d'interruption (Interrupt Register) est utilisé en liaison avec le mode d'interruption spécial IM3. Dans ce mode d'interruption, l'élément produisant l'interruption doit fournir, à la demande du processeur, une valeur 8 bits. Cette valeur comme low byte et le contenu du registre I comme high byte constituent l'adresse de la routine d'interruption.

Le registre R ou Refresh Register est utilisé en liaison avec le Refresh que le Z80 exécute automatiquement. Chaque fois qu'une instruction a été retirée, les sept bits inférieurs de ce registre sont automatiquement incrémentés. Le huitième bit reste toujours à 1 ou à 0, suivant sa

programmation.

Les registres I et R ne sont pas utilisés sur le CPC. Cependant, comme la valeur du registre R se modifie sans cesse, celui-ci peut être utilisé comme générateur de hasard.

1.2.3 Particularités du Z80 du CPC

Les nombreuses possibilités du Z80 laissent une grande marge de manoeuvre aux concepteurs de matériel ou de logiciel dans la construction d'un ordinateur. Cette CPU (unité centrale) peut être utilisée avec la même efficacité dans des systèmes très réduits ainsi que dans des machines aussi puissantes que le CPC.

Les développeurs du CPC se sont ingéniés à obtenir un maximum de puissance avec un minimum de composants. D'où certaines particularités qu'il est nécessaire de connaître pour pouvoir programmer et utiliser efficacement cette machine, particulièrement en langage-machine. Ce sont ces particularités que nous allons maintenant étudier.

Tout d'abord la gestion des interruptions du CPC.

La seule source d'interruptions du CPC est le gate array, ce composant fantastique qui contribue de façon décisive à la puissance de cet ordinateur. Toutes les 3,3 millisecondes, soit 300 fois par seconde, le gate array produit une brève impulsion qu'il place sur l'entrée IRQ* du Z80. L'entrée NMI* du processeur n'est pas utilisée et est disponible sur le connecteur d'extensions pour des extensions éventuelles.

La fréquence du signal d'interruptions est obtenue, à partir du signal H-Sync du CRTC 6845, au moyen d'un diviseur de fréquence. L'impulsion H-Sync qui apparaît environ toutes les 65 microsecondes est ici divisée par 52.

Comme le Z80 fonctionne sur le CPC en mode d'interruption IM1, chaque interruption IRQ identifiée provoque un RST7 ou encore un CALL &0038. Le processeur interrompt immédiatement le programme en cours, place l'état actuel du PC sur la pile et saute à l'adresse &0038. Ici figure, sur le CPC, un saut à l'adresse &B939 où se trouve la routine d'interruption proprement dite. Comme l'endroit où s'est produit l'interruption est enregistré sur la pile, le programme interrompu peut être repris une fois terminée la routine d'interruption.

Comme l'entrée IRQ* du processeur se trouve également sur le connecteur d'extension, on peut bien sûr se demander comment une interruption par le gate array peut être distinguée d'une interruption externe. Les

développeurs du CPC ont eu ici recours à une astuce. A l'intérieur de la routine d'interruption en &B939, l'interruption est à nouveau autorisée un court instant. Comme l'impulsion produite par le gate array ne dure pas plus de 5 microsecondes, cette autorisation de l'interruption n'a aucun effet, puisque l'impulsion est terminée depuis longtemps. Par contre, les sources externes d'interruption ne mettent fin à l'émission de leur signal que sur instruction expresse du processeur. Lorsqu'il y a une interruption externe, la routine d'interruption est donc elle-même interrompue. Ce cas peut être identifié et traité d'une manière spéciale. C'est ainsi que sont rendues également possibles les sources d'interruptions externes. La seule condition qu'elle doivent remplir, c'est une impulsion suffisamment longue.

Le second cas particulier qui doit être pris en compte, c'est la possibilité limitée d'utiliser les instructions de port.

En liaison avec le signal IORQ* (Input/Output ReQuest), le Z80 peut adresser un maximum de 256 ports différents, de façon analogue à l'adressage de cases mémoire. Pour cela, l'adresse du port souhaité est placée dans les 8 bits inférieurs d'adresse A0 à A7. Ces ports sont essentiellement utilisés pour connecter des éléments périphériques.

Sur d'autres processeurs qui ne connaissent pas l'adressage de port, le concepteur est toujours tenté d'adresser les éléments périphériques comme des cases mémoire. Ce procédé est appelé Memory Mapped et il présente l'inconvénient de réduire la zone d'adresses disponible pour la RAM.

Pour l'utilisation de l'adressage de port, le Z80 fournit le groupe très puissant des instructions IN et OUT. Si l'on étudie plus attentivement les instructions de ce groupe, on trouve dans les instructions IN(C),r et OUT(C),r une possibilité élégante d'adresser plus que les 256 ports normalement prévus. Dans ces instructions, l'état des 8 bits d'adresse inférieurs est déterminé par le contenu du registre C mais le contenu de B est en outre placé dans les bits d'adresse A8 à A15. C'est ainsi 65536 adresses de ports qui sont disponibles. C'est justement cette caractéristique du Z80 que les concepteurs du CPC ont utilisée. Tous les circuits intégrés périphériques sont sélectionnés au moyen des bits d'adresse A8 à A15.

De telles astuces ont malheureusement souvent un inconvénient. En l'occurrence l'inconvénient réside dans une nette limitation du jeu d'instructions du Z80. Aucune des autres instructions I/O du Z80 ne peut plus être utilisée. Ceci vaut notamment pour les instruction I/O avec automatisme de boucle. Ces instructions utilisent le registre B comme

compteur et ne peuvent donc pas 'fournir' le highbyte de l'adresse de port. C'est en particulier le cas des instructions INI, INIR, IND et INDR ainsi que OUTI, OTIR, OUTD et OTDR.

L'utilisation des cycles wait constitue une troisième particularité du CPC.

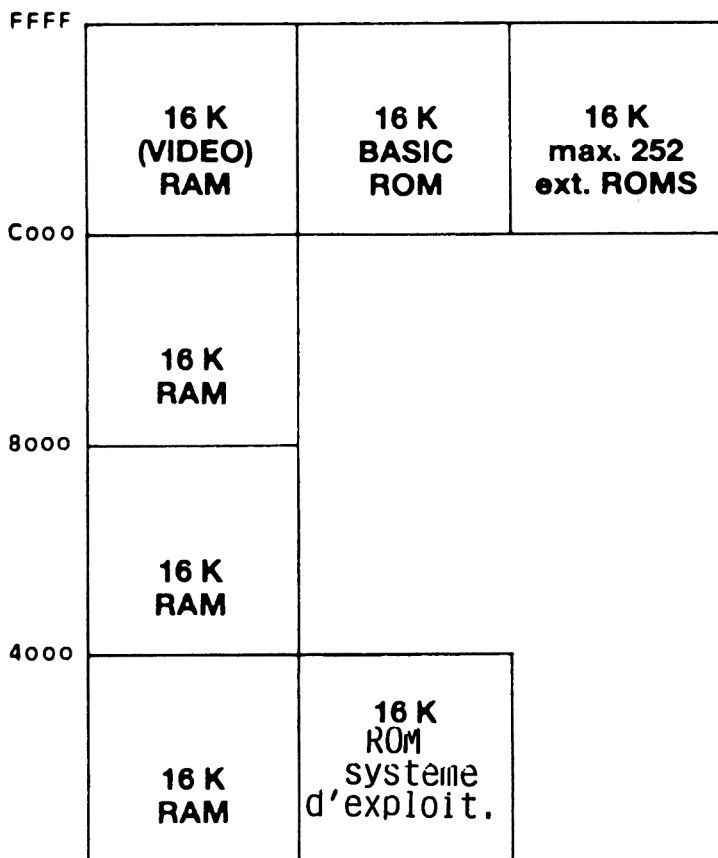
La nécessité de cette connexion du Z80 remonte à l'époque où les circuits intégrés de mémoire disponibles se la coulaient encore douce. Les premières EPROMs notamment n'étaient pas en mesure de préparer les données, après réception de l'adresse, avant un délai de quelques microsecondes.

Pour faire fonctionner le Z80 avec de tels 'paresseux', il fallait attendre un certain temps. Ce délai peut être produit par le signal WAIT*. Lors de chaque signal négatif sur l'entrée de l'horloge, le processeur examine l'état de la connexion WAIT*. Si cette connexion est à 0 Volt, le Z80 exécute ce que l'on appelle un cycle d'attente de la durée d'un mouvement d'horloge. Une fois écoulé le signal d'horloge, donc avec le signal négatif, l'état du canal WAIT* est à nouveau examiné, etc... L'utilisation de ce signal sur le CPC n'a cependant aucun rapport avec les circuits intégrés de mémoire utilisés. Ils sont tous suffisamment rapides pour un Z80 d'une fréquence de 4 MHz. La raison de l'utilisation de cette connexion est la nécessaire synchronisation entre processeur et contrôleur vidéo. Comme les deux circuits intégrés peuvent accéder à la mémoire, il faut contrôler de qui c'est le tour à un moment donné. Le contrôleur vidéo est d'ailleurs toujours prioritaire car sinon l'affichage sur le moniteur pourrait être sérieusement endommagé. Pour obtenir cette synchronisation, un signal WAIT* est produit pour le processeur tous les 4 mouvements d'horloge. Bien que le processeur fonctionne à 4 MHz (Méga Hertz= millions de vibrations par seconde), du fait des cycles d'attente, la fréquence de travail effective est d'environ 3,3 MHz.

Ce ralentissement de la vitesse de l'ordinateur n'est pas très grave en soi. Ce qui est plus gênant, c'est que les durée d'exécution des instructions correspondant aux données techniques fournies pour le processeur sont inexactes en ce qui concerne le CPC. C'est ainsi qu'il devient très difficile de réaliser des boucles de temporisation très précises telles qu'elles sont nécessaires par exemple pour utiliser des formats d'écriture sur cassette spéciaux et particulièrement rapides.

Les signaux BUSRQ* et BUSAK*, les signaux de commande du DMA ne sont pas

utilisés sur le CPC. Ils sont cependant placés sur le connecteur d'extension et sont donc disponibles pour des extensions externes. Le signal HALT*, qui n'est pas non plus utilisé sur le CPC est également disponibles sur le connecteur d'extension.



1.1.1.1. Organisation de la mémoire du CPC

1.3 Le gate array, le coordinateur du système

Presque tous les composants du CPC se trouvent couramment dans le commerce, dans n'importe quel magasin d'électronique bien approvisionné. Les seules exceptions sont la ROM et le gate array qui est désigné dans le schéma technique sous le nom de IC116. C'est ce dernier circuit intégré qui nous occupera dans cette section.

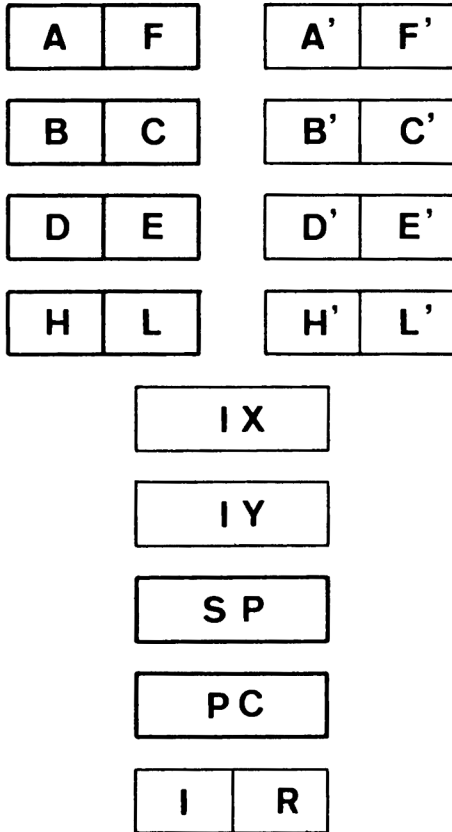
Ce circuit intégré à 40 pôles a été développé spécialement pour le CPC et il remplit plusieurs fonctions importantes. Si l'on voulait reconstituer toutes les fonctions intégrées avec des portes logiques TTL, le nombre de circuits intégrés ferait vite plus que doubler.

Les fonctions du gate array sont entre autres les suivantes:

- Production de toutes les fréquences d'horloge nécessaires
- Production des signaux pour l'exploitation de la RAM dynamique
- Commande des accès à la RAM
- Connexion et déconnexion de la ROM sur la zone de mémoire
- Production des signaux vidéo
- Production des informations RVB pour le moniteur couleur
- Commande du mode d'écran
- Stockage des couleurs d'encre
- Production de l'impulsion d'interruption

Il n'y a malheureusement que très peu d'informations disponibles sur ce circuit intégré très intéressant. Il est impossible d'obtenir une description technique de ce circuit intégré dont la vie interne est visiblement considérée par le constructeur comme un secret de fabrication.

Mais nos efforts et tentatives de découvrir le fonctionnement de ce circuit intégré de la façon la plus détaillée possible ont débouché sur un réel succès et nous ne voulons pas vous cacher les résultats auxquels nous avons abouti.



1.2.2.1. Jeu de registres Z80

1.3.1 L'affectation des pôles de connexion du gate array

Le signal qui détermine tout sur le CPC est le signal quartz d'une fréquence de 16 MHz qui se trouve sur le pin 8 (XTAL). Le IC125, un circuit intégré TTL du type 7400, constitue avec deux de ses quatre portes logiques une commutation d'oscillateur typique. Ce signal constitue pratiquement le battement cardiaque du CPC.

La fréquence d'entrée divisée par quatre est disponible pour le processeur, sous la forme d'un signal d'horloge de 4 MHz sur le pin 39 comme mouvement d'horloge

Une nouvelle division par quatre donne une fréquence de 1 MHz. Ce signal est fourni sur le pin 1 du gate array.

Le signal de 1 MHz a deux emplois. C'est tout d'abord le signal d'horloge pour le chip sonore et il contribue ensuite à déterminer si le processeur ou le CRTC peut adresser la RAM. S'il y a un low, les canaux d'adresse du processeur sont commutés sur la RAM à travers les circuits intégrés multiplexeurs IC 104, 105, 109 et 113.

Comme par ailleurs la commande de la RAM sur le CPC n'est pas tout à fait évidente, vous trouverez une description détaillée des signaux de commande de la RAM dans un prochain chapitre.

Comme les composants de mémoire ne disposent que de 8 canaux d'adresse, l'adresse totale de 16 bits doit être multiplexée, c'est-à-dire placée sur les entrées avec un décalage dans le temps. Cette commande dans le temps est obtenue avec les signaux CAS ADDR*(pin 6), CAS*(pin 3) et RAS*(pin 7). Ces signaux RAS* et CAS* sont placés directement vers les RAMs, le signal CAS ADDR* est conduit vers les multiplexeurs que nous avons déjà évoqués.

Le signal MAO/CCLK sur le pin 40 du gate array a également une fréquence de 1 MHz. Ce signal est par ailleurs déphasé par rapport au signal CPU ADDR*, c'est-à-dire que les deux fréquences sont high à des moments différents. MAO/CCLK a également une double fonction. Il constitue d'une part le signal d'horloge pour le CRTC qui tire tous les autres signaux de ce signal; d'autre part il est placé comme bit d'adresse auxiliaire sur le multiplexeur IC 106. La fonction de ce bit d'adresse auxiliaire sera

également évoquée plus tard plus précisément, à propos de la commande de la RAM.

Le gate array produit encore sur le pin 13 le signal RAMRD*. Cette connexion devient low, lorsque le processeur, après avoir fourni une adresse, veut lire des données dans la RAM et qu'il l'indique au gate array par son signal RD* sur le pin 19. Comme la ROM et la RAM se chevauchent sur de grandes zones, le signal RD* du processeur ne peut être utilisé directement. Si des données doivent être lues dans la ROM, le signal RAMRD* reste high et les sorties du IC114, qui est ce qu'on appelle un buffer (mémoire provisoire) deviennent high. Dans ces moments, aucune information ne peut passer de la RAM sur le bus de données, bien que l'adresse de la mémoire soit également parvenue à la RAM et que celle-ci tienne un octet prêt dans ses sorties.

En plus du RAMRD*, le signal READY du pin 2 du gate array est placé sur l'IC114. Ce signal produit sur le processeur le signal pour l'intégration des cycles d'attente. La liaison supplémentaire entre le READY et l'IC114 permet d'obtenir que l'information sur le bus de données du processeur ne se modifie pas pendant les cycles d'attente. Le 74LS373 stocke, après envoi d'un high, l'information en sortie actuelle sur le pin 11, jusqu'à ce que ce pôle devienne low. Le circuit intégré se comporte ensuite comme un simple buffer, c'est-à-dire que les sorties suivent immédiatement les modifications des entrées.

Le signal ROMEN* sur le pin 12 du gate array devient low lorsque le processeur veut lire des données dans la ROM. La ROM intégrée de 32 K du CPC occupe les zones d'adresses &0000 à &3FFF et &C000 à &FFFF. Cette ROM peut donc être appelée en deux moitiés distinctes. Dans les zones de mémoire où RAM et ROM se chevauchent, il faut indiquer au gate array le choix fait avec une instruction OUT. Il est ainsi tout à fait possible de n'activer qu'une moitié de la ROM.

Conformément à la configuration de la mémoire choisie, le gate array décode l'état des canaux d'adresse A14 et A15. Suivant la mémoire demandée c'est le signal RAMRD* ou ROMEN* qui sera activé lors de la lecture.

Une instruction d'écriture du processeur va toujours vers la RAM, indépendamment de la configuration de la mémoire choisie. Le gate array produit à cet effet le signal MWE*.

Outre la fonction décrite, les canaux d'adresse A14 et A15 sur les pins 20 et 21 sont encore utilisés dans un autre but. Le gate array a une adresse de port qui est utilisée pour programmer les différentes possibilités du gate array. L'adresse de port est &7F00 et elle est décodée sur le pin 18, à travers les canaux d'adresse (A14 High, A15 Low) et le signal IORQ*.

Comme le bus de données du Z80 n'est pas directement relié aux canaux de données D0 à D7 du gate array, le GA (gate array) met le pôle 244EN* sur low lorsque l'adresse de port &7F00 est identifiée de la façon que nous avons indiquée. Les sorties de l'IC115 (74LS244, un buffer de bus de données) sont ainsi libérées et l'octet fourni par le Z80 peut être écrit dans le GA.

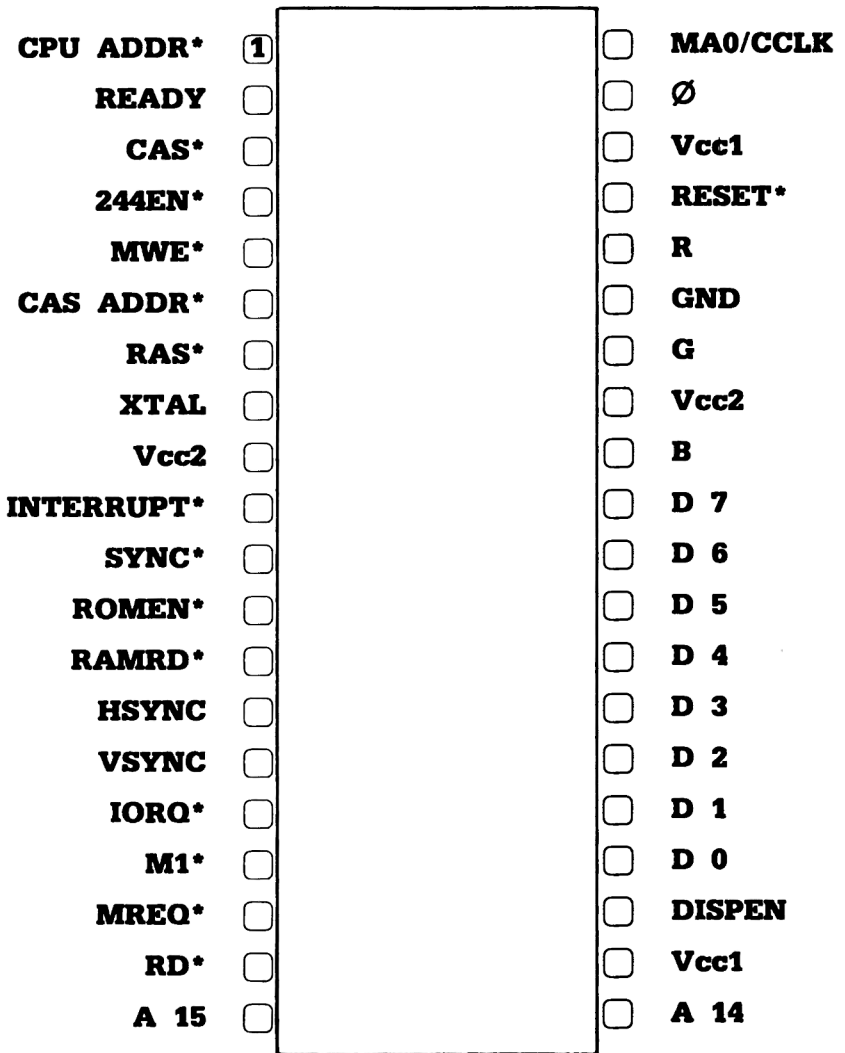
Mais le signal IORQ* a lui aussi une double signification pour le GA. Le Z80 a en effet la particularité, lorsqu'il identifie une interruption, de mettre simultanément à low les signaux IORQ* et M1*. Cette situation est identifiée par le GA et l'impulsion d'interruption est immédiatement annulée. Si par contre, le traitement de l'IRQ a été interdit par l'instruction DI, Disable Interrupt, le pôle 10 du GA reste low, jusqu'à ce que l'IRQ soit à nouveau autorisé. Dès que l'IRQ est à nouveau autorisé par l'instruction EI, Enable Interrupt, l'interruption présente est identifiée et la sortie d'interruption redevient high.

Le signal d'interruption sur le pin 10 est produit par une chaîne de division programmable du GA. Cette chaîne de division est alimentée par le signal HSYNC du CRTC et elle divise la fréquence existante par 52. Comme l'impulsion HSYNC se produit environ toutes les 65 microsecondes, l'intervalle entre deux impulsions d'interruption est de 3,3 millisecondes. Les impulsions sont couplées avec le signal VSYNC du CRTC. La durée du VSYNC est programmée dans le CRTC à environ 500 microsecondes. Après environ 125 microsecondes apparaît l'interruption, de sorte que la routine d'interruption a encore environ 375 microsecondes pour examiner sur le bit 0 du port B du 8255 s'il y a un VSYNC. Ce signal est utilisé comme horloge dans différentes opérations.

Ce cas ne se produit cependant que toutes les 15 interruptions, pour les 14 interruptions restantes, il y a un high du VSYNC et le compteur interne n'est pas affecté.

Mais les signaux HSYNC et VSYNC sont bien sûr nécessaires, de même que DISPEN pour produire le signal vidéo. Une liaison de ces signaux donne le

signal SYNC* sur le pin 11 du GA.



1.3.1.1. Pinout du Gate Array

1.3.2 La structure des registres du gate array

L'exécution de toutes les tâches que nous avons décrites nécessite que les données soient stockées dans le GA. Le nombre exact des registres internes n'est pas connu mais nous pensons pouvoir décrire les registres les plus importants.

Comme tous les autres éléments du CPC, le GA est appelé à travers l'adressage de port.

Il occupe l'adresse &7Fxx. Il en résulte donc que le bit d'adresse A15 doit être low et le bit d'adresse A16 high. Les autres bits d'adresse (A12 à A18) doivent être mis (sur le niveau high) puisque les autres éléments périphériques sont décodés d'une manière semblablement incomplète. Sur ces périphériques, les entrées de sélection ne sont également reliées qu'aux différents bits d'adresse.

L'état de l'octet d'adresse inférieur est sans importance pour le décodage et n'importe quelle valeur peut y figurer.

On peut distinguer en tout trois différents registres.

Les deux premiers registres sont liés à la production des couleurs, plus précisément aux affectations de couleur fixées avec PEN et INK.

Le premier registre reçoit l'adresse dans laquelle la valeur de couleur doit être écrite. Nous le désignerons désormais sous le nom de registre du numéro de couleur (reg NC).

La valeur de la couleur elle-même peut être ensuite écrite dans le second registre (sous la même adresse de port!). Nous appellerons ce registre registre de valeur de couleur (reg VC).

Le troisième registre est un registre multi-fonctions (reg MF) qui détermine le mode d'écran et la configuration de la mémoire. La sélection des différentes possibilités y est déterminée par les différents bits à l'intérieur du registre.

Dans tous les registres du GA, il est seulement possible d'écrire. Il est IMPOSSIBLE de lire les valeurs de ces registres.

Comme le GA ne peut être appelé qu'à travers une seule adresse de port, il faut qu'il y ait un moyen de distinguer les différents groupes. Cette distinction est opérée grâce aux deux bits supérieurs de l'octet de donnée. Les combinaisons possibles sont:

Bit 7	Bit 6	
0	0	Ecrire une valeur dans le reg NC
0	1	Ecrire une valeur de couleur dans le reg VC choisi
1	0	Ecrire une valeur dans le reg MF
1	1	Inutilisé?

Mais que représentent les registres de numéro de couleur et de valeur de couleur?

Fondamentalement, ces registres correspondent aux instructions PEN et INK. L'instruction PEN modifie la couleur d'écriture actuelle sur le moniteur. L'affectation d'un numéro PEN à une couleur peut être fixée avec l'instruction INK. Il faut pour cela indiquer le numéro à modifier et la valeur souhaitée. Ce sont exactement ces fonctions qu'exécutent ces deux registres. Le numéro de la couleur à modifier est placé dans le registre NC, après quoi la valeur de couleur souhaitée est écrite dans le registre VC.

Pour modifier par exemple la couleur affectée à PEN 1, il faut employer les instructions suivantes:

```
OUT &7F00,&X0000001:OUT &7F00,&X010XXXX
```

Dans la première instruction OUT, les bits 6 et 7 valent 0 et les bits 0 à 3 contiennent le numéro de la couleur à modifier. Dans notre exemple, il s'agit du numéro 1. Le bit 5 n'a pas de fonction, le bit 4 a une fonction spéciale sur laquelle nous reviendrons bientôt.

Dans la seconde instruction OUT, les bits 6 et 7 ont été choisis de façon à ce que le registre VC soit sélectionné. Les bits 'X' correspondent simplement à la valeur de couleur. 5 bits permettent en principe de sélectionner 32 couleurs différentes mais il n'y a que 27 couleurs différentes. Les 5 valeurs de couleur restantes sont identiques à d'autres couleurs.

Si vous essayez cet exemple en Basic, vous constaterez que le succès escompté se fait attendre. Tout ce que vous obtenez, c'est un rapide flash de la nouvelle couleur.

La cause en est une particularité du logiciel du CPC. Toutes les couleurs sont représentées en "clignotement". Vous ne le remarquez pas parce que le clignotement ne se fait pas entre couleurs différentes, mais entre couleurs identiques. Lors de chaque commutation entre deux couleurs, tous les paramètres pour le GA sont chargés à nouveau. Mais si, avant les

instructions OUT, vous utilisez l'instruction SPEED INK 255,255, vous pourrez observer nettement plus longtemps, au moins lors de quelques tentatives, l'effet de ces instructions.

Venons-en maintenant à l'explication du bit 4 du reg NC. Si ce bit est lors de l'accès fixé sur le registre, l'information des bits 0 à 3 sera ignorée et la valeur de couleur transmise par la prochaine instruction OUT sera interprétée comme nouvelle couleur du bord.

Le registre MF est adressé lorsque, dans l'instruction OUT, le bit 7 est mis et le bit 6 est low. Les autres bits de ce registre ont la signification suivante:

- Bit 5: Aucune fonction?
- Bit 4: 1 = annuler le compteur V Sync
- Bit 3: 1 = déconnecter ROM &C000 à &FFFF
- Bit 2: 1 = déconnecter ROM &0000 à &3FFF
- Bit 1: Mode écran
- Bit 0: Mode écran

Nous n'avons rien pu découvrir jusqu'ici sur la fonction du bit 5.

Si le bit 4 est mis, la chaîne de division pour l'impulsion d'interruption est annulée et le processus de comptage des impulsions V Sync recommence du début. Il serait ainsi possible d'allonger l'intervalle entre deux impulsions d'interruption. Vous pouvez constater cette fonction en Basic grâce à la boucle de programme suivante:

```
10 OUT &7F00,&X10010110:GOTO 10
```

Après avoir lancé cette ligne de programme, vous constatez que l'ordinateur est complètement bloqué et qu'un RESET avec SHIFT/CTRL/ESC n'est même plus possible. Cette ligne provoque en effet une annulation si rapide du registre de comptage, que plus aucune impulsion d'interruption ne peut plus se produire. Et comme le clavier est interrogé par la routine d'interruption, vous ne pouvez plus réutiliser votre CPC qu'après l'avoir éteint puis rallumé.

Les bits 2 et 3 déterminent la configuration de la mémoire actuelle. Si l'un des bits est mis, c'est la Ram que le processeur rencontrera dans la

zone d'adresse correspondante, lors de ses accès en lecture, si ces bits sont nuls, le processeur lira des données dans la Rom.

Une manipulation désordonnée de ces bits débouche au minimum sur des messages d'erreur mais le "plantage" complet du système ou un Reset sont également possibles.

Les bits restants 0 et 1 déterminent le mode actuel de l'écran. Les combinaisons possibles sont:

Bit 1	Bit 0	
0	0	Mode 0, 20 colonnes, 16 couleurs
0	1	Mode 1, 40 colonnes, 4 couleurs
1	0	Mode 2, 80 colonnes, 2 couleurs
1	1	Comme Mode 0, mais sans clignotement

Si vous avez essayé notre programme d'une ligne pour supprimer les interruptions en mode 1, vous aurez certainement constaté une très curieuse modification des caractères sur l'écran. Dans cet exemple, nous avons choisi comme mode écran le mode 80 colonnes et changé de mode sans vider l'écran. Les caractères représentés se présentent comme s'il manquait des points au milieu de chaque caractère. Vous trouverez l'explication de ce phénomène à la fin du chapitre suivant, lorsque nous décrirons la structure de l'écran et la représentation des caractères.

1.4 Le contrôleur vidéo HD 6845

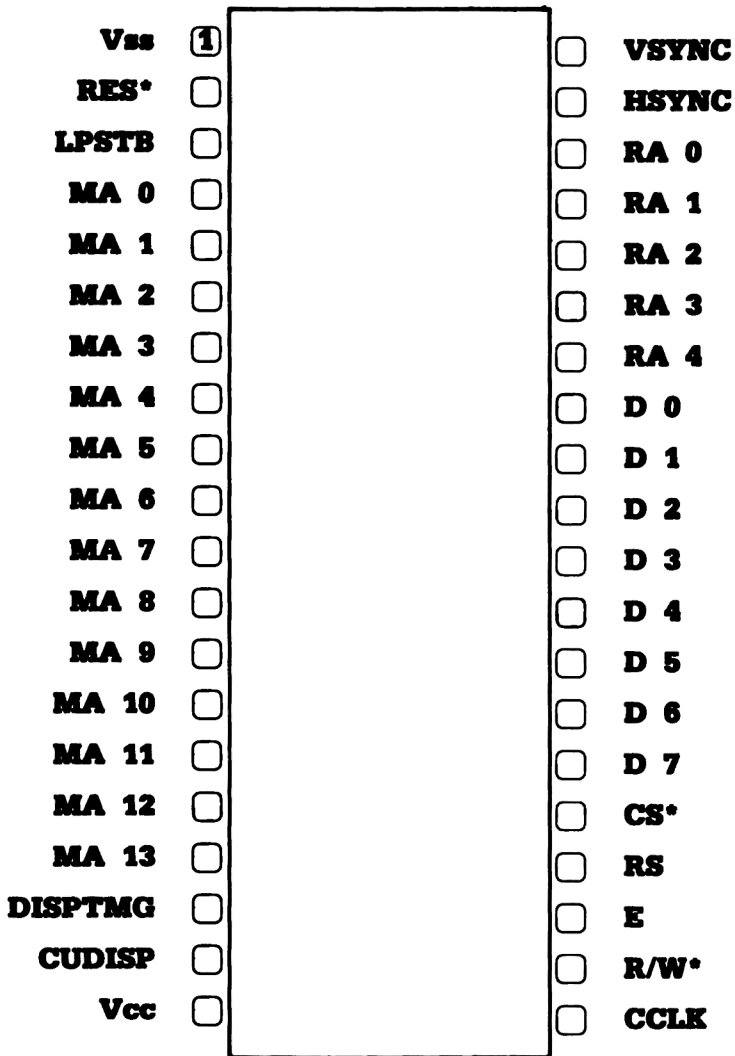
Le travail principal dans la production de l'image sur le moniteur est accompli par le contrôleur vidéo HD 6845, également désigné comme Cathode Ray Tube Controller, CRTC en abrégé. Ce circuit intégré a été spécialement conçu comme une interface entre des microprocesseurs et des écrans à grille tels que les moniteurs courants.

Il produit, à partir d'un signal d'horloge unique, tous les signaux de synchronisation nécessaires pour le moniteur et tous les paramètres nécessaires à cet effet peuvent être programmés à l'intérieur de limites assez larges.

Avant de décrire l'affectation des pîes de connexion et la structure interne de registres, nous voulons vous donner un bref aperçu des possibilités de cet élément intéressant:

- Nombre de caractères par ligne programmable
- Nombre de lignes par écran programmable
- Matrice de points verticale des caractères programmable
- Accès à une zone de mémoire de 16 K
- Refresh automatique pour l'utilisation de Rams dynamiques
- Fonctions de contrôle du curseur
- Curseur programmable (hauteur et clignotement)
- Entrée light-pen
- Alimentation en 5 volt continu
- Entrées/Sorties compatibles TTL

Le 6845 fut développé à l'origine par Motorola pour être employé dans des systèmes informatiques dotés de processeurs de la famille 68xx. Mais du fait de son extraordinaire flexibilité et de sa manipulation aisée ce contrôleur se rencontre sur de très nombreux systèmes, y compris sur des systèmes aussi puissants que par exemple Sirius.



1.4.1.1. Pinout du CRTIC HD 6845

1.4.1 Les pôles de connexion du CRTC

La signification des 40 pattes de connexion est la suivante:

- MAO-13 : Memory Adress Lines; les cases mémoire de la mémoire écran sont adressées à travers ces 14 connexions
- RAO-4 : Raster Adress Lines; ces 5 connexions choisissent à partir du générateur de caractères la ligne actuelle de la grille du caractère à représenter
- DO-7 : Bidirectional Data Bus; des informations peuvent être écrites dans le contrôleur et lues à partir de celui-ci à travers ces pins
- R/W* : Read/Write*; ce signal détermine le sens des données sur les canaux de données. Avec un low, les données peuvent être écrites du processeur dans le CRTC, avec un high, elles sont lues à partir du CRTC.
- CS* : Chip select*. Pour permettre des transferts de données avec le 6845, celui-ci doit être adressé, ce qui est obtenu par un low sur l'entrée CS*.
- RS : Register Select. Ce signal est utilisé pour choisir entre le registre d'adresse et 18 registres de contrôle. Avec un niveau low sur RS, on peut accéder au registre d'adresse, avec un high, on accède au registre de contrôle.
- EN : Enable. Avec une bascule ascendante de ce signal, les signaux du processeur se trouvant sur le circuit intégré sont pris en compte.
- RES* : Reset*.Adress Lines; les cases mémoire de la mémoire écran sont adressées à travers ces 14 connexions
- CLK : Character Clock est le signal d'horloge dont sont tirés par division tous les signaux dont a besoin le moniteur.
- HSYNC : Horizontal Sync fournit le signal de synchronisation horizontale du moniteur. La mauvaise définition ou l'absence de HSYNC se traduit par un défilement de l'image.
- VSNC : Vertical Sync fournit le signal de synchronisation verticale du moniteur.
- DISPTMG : Display Timing. Ce signal est high lorsque le signal envoyé au moniteur doit être représenté à l'écran. Ce signal permet d'inhiber les retours en arrière du faisceau
- CUREN : Cursor Enable (souvent appelé également Cursor Display ou CURDISP) est utilisé lorsque le curseur n'est pas commandé par

le logiciel mais par le CRTC lui-même. Cette connexion permet également de commander le clignotement du curseur.

LPSTB : Light Pen Strobe. Si une bascule low-high est envoyée sur cette entrée, l'état actuel des canaux MA est transféré et stocké dans les registres Light-pen. Ces registres peuvent être lus pour être utilisés dans un programme.

1.4.2 Les registres internes du contrôleur vidéo

Comme nous l'avons déjà indiqué, le 6845 contient un registre d'adresse et 18 registres de contrôle. Comme le signal RS, Register Select, ne permet toutefois de choisir qu'entre deux adresses, on peut donc se demander comment il est possible d'appeler les 18 registres de contrôle à travers une seule adresse. La solution de ce problème réside dans le registre d'adresse. Le numéro du prochain registre de contrôle auquel on veut accéder est écrit dans le registre d'adresse. Ce procédé semble certes relativement compliqué mais il présente un avantage indéniable. De cette façon en effet, le CRTC n'occupe justement que deux adresses et non pas 18 ou même 32. Comme d'autre part le CRTC n'est normalement programmé qu'une seule fois, lors de la mise sous tension de la machine, ce travail de programmation supplémentaire est tout à fait acceptable.

Mais examinons maintenant les 18 registres un peu plus en détail. La description suivante semblera peut-être un peu sèche et difficilement compréhensible à cause de la structure complexe des différents registres. De même, certaines connaissances de base en technique vidéo sont nécessaires pour la compréhension de certains registres. Si vous ne comprenez pas tout à la lecture de cette description, consolez-vous en vous disant que le contrôleur vidéo de votre ordinateur n'a absolument pas à être programmé "manuellement".

Dans la présentation suivante, un R placé à la suite du nom du registre indique que ce registre doit être lu et un W signifie qu'on peut écrire dans ce registre. Remarquez que certains registres peuvent uniquement être lus ou écrits, ce qui est indiqué par -.

AR -/W : Adress Register. Ce registre 5 bits reçoit le numéro du registre de contrôle souhaité. Les valeurs de registre 18 à 31 sont ignorées, les seules valeurs valables vont de 0 à 17. Ce

registre est appelé lorsqu'aussi bien CS que RS sont low.

- R0 -/W : Horizontal Total. Ce registre 8 bits reçoit le nombre de caractères par ligne complète. Notez d'ailleurs qu'une ligne complète est nettement plus grande que les caractères visibles à l'écran car les durées pour le bord et le retour en arrière du faisceau doivent être également prise en compte dans le calcul. Cette valeur est donc environ 1 fois et demi plus importante que le nombre choisi de caractères par ligne.
- R1 -/W : Horizontal Displayed. Ce registre contient le nombre de caractères à représenter à l'écran. La valeur placée ici doit être inférieure à celle de R0.
- R2 -/W : Adress Register. Ce registre 5 bits reçoit le numéro du registre de contrôle souhaité. Les valeurs de registre 18 à 31 sont ignorées, les seules valeurs valables vont de 0 à 17. Ce registre est appelé lorsqu'aussi bien CS que RS sont low.
- R3 -/W : Sync Width. Les 4 bits inférieurs de ce registre déterminent la largeur des impulsions HSync et VSync. Les 4 bits supérieurs de ce registre ne sont pas utilisés.
- R4 -/W : Vertical Total. Les 7 bits inférieurs de ce registre déterminent le nombre total de lignes de grille par image. Cette valeur détermine donc ainsi également si la fréquence de renouvellement de l'image est de 50 ou 60 Hertz.
- R5 -/W : Vertical Total Adjust. Les 6 bits inférieurs de ce registre permettent de réaliser un ajustement précis de la fréquence de renouvellement de l'image.
- R6 -/W : Vertical displayed. Les 7 bits inférieurs de ce registre déterminent le nombre de lignes de grille réellement représentées sur le moniteur. Théoriquement, on peut programmer ici n'importe quelle valeur inférieure au contenu de R4.
- R7 -/W : Vertical Sync Position. La valeur 7 bits de ce registre détermine le moment de l'impulsion VSync. Si la valeur de R7 est diminuée, l'image du moniteur est alors décalée vers le

bas, si cette valeur est augmentée, il y a décalage vers le haut.

R8 -/W : Interlace. Les deux bits inférieurs de ce registre permettent de déterminer si la représentation doit avoir lieu avec ou sans procédure de saut de ligne (interlace).

R9 -/W : Maximum Raster Adress. Ce registre 5 bits détermine le nombre de lignes de grille des caractères à représenter.

R10 -/W : Cursor Start Raster. Les bits 0 à 4 de ce registre déterminent sur quelle ligne de la grille doit commencer le curseur. Les bits 5 et 6 fixent le mode de curseur de la façon suivante:

Bits	6	5	
	0	0	Curseur non clignotant
	0	1	Curseur non représenté
	1	0	Curseur clignotant (env. 3 par seconde)
	1	1	Curseur clignotant (env. 1.5 par seconde)

R11 -/W : Cursor End Raster. En fonction du contenu de R10, les 5 bits inférieurs de ce registre déterminent sur quelle ligne de l'écran se termine le curseur.

R12 R/W : Start Adress High. Les bits 0 et 5 déterminent à partir de quelle adresse de tout le domaine d'adressage de 16 K du CRTC commence la mémoire écran. Si ce registre est lu, les bits 6 et 7 sont toujours low.

R13 R/W : Start Adress Low. Ce registre fixe, de façon analogue à R12 l'octet d'adresse faible de la mémoire écran à adresser.

R14 R/W : Cursor High. Les bits 0 à 5 de ce registre représentent l'octet fort de la position actuelle du curseur.

R15 R/W : Cursor Low. De façon analogue à R14, ce registre reçoit l'octet faible de l'adresse du curseur.

Comme R14 ainsi que R15 peuvent être écrits ou lus, ces

registres permettent de déterminer librement la position du curseur.

R16 R/- : Ce registre contient après une impulsion strobe positive l'octet fort de l'adresse de la mémoire écran qui était activée au moment de l'impulsion. Les bits 6 et 7 de ce registre sont toujours low.

R17 R/- : De façon analogue à R16, ce registre contient l'octet faible au moment du strobe light-pen.
R16 ainsi que R17 ne peuvent qu'être lus.

1.5 La Ram du CPC

La RAM (mémoire écriture/lecture) de 64 K intégrée dans le CPC n'est pas uniquement utilisée comme mémoire de donnée et de programme. Les informations concernant l'écran sont également placées dans cette mémoire.

Après que nous ayons étudié en détail dans les chapitres précédents les trois éléments les plus importants, le processeur, le gate array et le contrôleur vidéo, nous allons dans le présent chapitre jeter un regard sur l'interaction de ces trois éléments lors de l'accès aux circuits intégrés de mémoire. Nous expliquerons également à cette occasion comment le contrôleur vidéo appelle la Ram pour représenter des caractères à l'écran.

Mais nous voulons faire auparavant une petite digression pour étudier comment fonctionnent les éléments de mémoire.

Nous allons tout d'abord expliquer comment est possible l'adressage de 65536 cases mémoire avec les 8 connexions d'adresse existantes.

Le principe consiste à diviser l'adresse 16 bits en deux moitiés et à envoyer ces deux octets d'adresse l'un après l'autre sur les pins d'adresse de la Ram. Ce procédé est appelé multiplexage. Le multiplexage nécessite cependant des signaux qui indiquent à la Ram quelle information se trouve dans l'instant sur les connexions d'adresse.

C'est ici qu'entrent en jeu les signaux RAS* et CAS* fournis par le gate array.

Après qu'un octet d'adresse ait été envoyé aux Rams, une bascule high-low du signal RAS* leur indique qu'une moitié d'adresse est prête. Avec la bascule négative (high-low) du RAS*, l'information d'adresse disponible est stockée dans les Rams.

La deuxième moitié de l'adresse peut alors être envoyée à la Ram. Dès que cet octet d'adresse est prêt, le signal CAS* devient low. La Ram a ainsi reçue la totalité de l'adresse 16 bits et sélectionne alors la case mémoire souhaitée. Il est maintenant possible d'écrire ou de lire cette case.

La commutation des moitiés d'adresse doit bien sûr être également prise en charge par un signal convenable, sur le CPC, c'est le signal CAS-ADDR*.

Le multiplexage est effectué par les circuits intégrés IC 104, 105, 109 et 113. On peut se représenter le fonctionnement de ces circuits intégrés du type 74LS153 comme deux commutateurs commandés électroniquement. A travers deux entrées de commande, on peut décider laquelle des quatre entrées doit être reliée à la sortie.

Les deux entrées de commande sont commandées par les signaux CPU-ADDR* et CAS-ADDR*. Le signal CPU-ADDR* permet de décider si c'est le processeur ou le CRTC qui peut envoyer une adresse à la Ram, et CAS-ADDR* effectue la commutation entre les moitiés d'adresse.

Nous allons examiner un exemple de commutation avec le multiplexeur IC 105.

Les pins de sortie 7 et 9 sont reliés chacun à travers une résistance de 120 Ohms avec les entrées d'adresse A0 et A1 des Rams.

Les entrées de commande A (pin 14) et B (pin 2) sont reliées aux signaux CPU-ADDR* et CAS-ADDR* que nous connaissons.

L'information d'adresse se trouve sur les pins 3 à 6 et 10 à 13. C'est ici aussi que vous retrouvez le signal CCLK que nous avons qualifié au chapitre précédent de bit d'adresse auxiliaire. Le tableau suivant indique quel bit d'adresse apparaît sur les sorties, suivant la combinaison de commande:

CPU- ADDR	CAS- ADDR	SORTIE A0 DU MULTIPLEXEUR	SORTIE A1 DU MULTIPLEXEUR
0	0	Z80,A9	Z80,A0
0	1	Z80,A2	Z80,A1
1	0	6845,MA8	CCLK
1	1	6845,MA1	6845,MA0

A première vue, ce tableau ne contribue pas particulièrement à la compréhension de la commande des Rams. Il est particulièrement troublant que le canal d'adresse A0 du processeur ne se trouve pas sur A0 de la Ram. Considérez cependant qu'il est indifférent au processeur de savoir dans quelle adresse physique de la Ram il écrit son information. Il est par exemple sans aucune importance pour le processeur que lorsqu'il écrit ou lit 'sa' case mémoire 0, ce soit vraiment l'adresse physique 0 de la Ram qui soit adressée ou que ce soit une tout autre adresse. De toute façon, chaque fois qu'il accédera à 'sa' case mémoire 0, c'est toujours la même case mémoire qui sera adressée. La désignation des pins d'adresse de la Ram est donc arbitraire et sans importance pour le processeur.

Baucoup plus importante est l'affectation d'adresses du processeur aux adresses du CRTC. Cette affectation est montrée par la figure 1.5.0.1.

Comme on voit, tous les bits d'adresse du processeur sont envoyés à travers les multiplexeurs sur les connexions d'adresse des Rams mais le contrôleur vidéo adresse également avec l'aide du CCLK l'ensemble de la zone adressable de 64 K. Ce qui contredit cependant le chapitre précédent où nous disions que le CRTC ne peut adresser qu'une zone de 16 K. Cette affirmation était exacte dans la mesure où seules les 14 connexions désignées par MA (Memory Adress Line) peuvent être comptées comme canaux d'adresse. Ces 14 connexions permettent d'adresser une zone d'adresse de 16 K.

Z80	6845	Z80	6845
A0	CCLK	A8	MA7
A1	MA0	A9	MA8
A2	MA1	A10	MA9
A3	MA2	A11	RA0
A4	MA3	A12	RA1
A5	MA4	A13	RA2
A6	MA5	A14	MA12
A7	MA6	A15	MA13

1.5.0.1 Accès du Z80 et du 6845 à la mémoire commune

Le mode de travail du 6845 utilisé dans le CPC pour l'adressage de la mémoire vidéo est rarement employé. Les connexions RA0 à RA4 servent normalement à appeler une Rom de caractères déjà programmée qui contient le modèle bits des caractères qui doivent être représentés à l'écran.

Les ordinateurs ont normalement une zone de mémoire appelée mémoire vidéo dans laquelle sont stockés tous les caractères à représenter à l'écran. Dans cette mémoire, l'emplacement de chaque caractère occupe un octet. Cela donne donc, pour représenter 80 x 25 caractères, une mémoire de 2000 octets.

Mais il n'est pas possible de faire entrer dans un octet toutes les informations nécessaires pour la représentation des caractères. Chaque caractère se compose en effet d'un certain nombre de lignes de points placées les unes sous les autres.

Sur le CPC, on peut également reconnaître ces lignes sur le moniteur. C'est ainsi par exemple que le curseur se compose de 8 lignes placées les unes sur les autres, dont tous les points image sont "allumés". Pour représenter des lettres ou des chiffres, seuls les points d'une ligne correspondant à la forme de la lettre ou du chiffre sont allumés. Les modèles de ces lignes de points sont stockées sous forme de cartes bits, dans lesquelles un bit mis correspond normalement à un point allumé sur l'écran.

Les connexions RA sont maintenant nécessaires pour recevoir de la Rom de caractères les différentes lignes, c'est-à-dire les cartes bits. A cet effet, les connexions RA sont utilisées comme canaux d'adresse pour la

Rom de caractères.

Comme vous pouvez l'imaginer, il n'est pas possible de réaliser à l'écran du graphisme haute résolution lorsqu'on utilise une Rom de caractères. Les ordinateurs fonctionnant suivant ce principe ne peuvent sortir du jeu de caractères intégré.

Sur le CPC, cette Rom de caractères n'existe pas et on a choisi une voie totalement différente.

Comme les connexions RA adressent directement la mémoire, les informations sur les points doivent donc nécessairement figurer également en Ram. Ce n'est qu'à travers cette astuce de commutation qu'il est possible de produire n'importe quelle carte bits sur le moniteur et donc de représenter le graphisme dans les limites connues.

Mais avant que nous ne nous tournions vers la structure concrète de la mémoire vidéo, il nous faut enfin expliquer le signal CCLK. Mais il nous faudra pour cela un peu de mathématiques.

Le CRTC est commandé par une fréquence d'horloge de 1 MHz. Avec chaque impulsion d'horloge est adressée une case mémoire. Dans cette case se trouvent les informations sur les points qui doivent être représentés 'allumés' sur l'écran, c'est-à-dire dans la couleur d'écriture. Comme une fréquence de 1 MHz correspond à une période de 1 micro-seconde, exactement un huitième de la fréquence d'horloge est disponible pour la représentation de chaque point, soit 0.125 micro-secondes. Pour représenter les 640 points d'une ligne, il faut donc un temps de 80 micro-secondes.

Mais comme le signal V Sync qui détermine la durée d'une ligne a une période de 52 micro-secondes, le compte n'est pas bon. Ces valeurs ne permettent de représenter que 40 caractères au maximum.

La solution à ce problème réside dans un mode spécial de travail des Rams, le Page Adress-Mode (mode d'adressage par page). Lorsqu'une Ram, après avoir envoyé les signaux RAS et CAS, envoie le contenu de la case mémoire souhaitée sur les sorties de donnée, il suffit alors de n'envoyer avec une autre impulsion CAS qu'une nouvelle moitié d'adresse aux Rams pour recevoir l'octet suivant. Cela suppose bien sûr que seule une moitié des informations d'adresse ne change.

C'est exactement cette possibilité qu'ont utilisée les développeurs du CPC. Bien sûr, il faut que les informations d'adresse correspondant aux deux différentes impulsions CAS soient différentes, sinon on lit deux fois la même case mémoire. Mais c'est justement ce que réalise le signal CCLK qui commute exactement entre les deux impulsions CAS. Ce signal est

envoyé par le multiplexeur IC 105 sur le bit d'adresse 0 (du point de vue du processeur), lorsque le signal CAS-ADDR est sur low et le signal CPU-ADDR par contre sur high. Ce signal représente bien ainsi le bit d'adresse inférieur de la Ram vidéo.

Les deux octets fournis rapidement l'un après l'autre par la Ram vidéo sont entrestockés dans le gate array, convertis dans la forme sérielle indispensable pour le moniteur et envoyés avec les informations de couleur sur la sortie RVB.

Restent encore les deux signaux MA12 et MA13. Ces deux signaux permettent de déterminer par blocs de 16 K le début de la Ram vidéo. Ces bits sont normalement mis et la Ram vidéo commence donc en &C000. Mais il est également possible d'obtenir par programmation que la Ram vidéo soit placée de &4000 à &7FFF.

1.6 La Ram vidéo entre 280 et 6845

Essayez maintenant ce petit programme sur le CPC:

```
10 MODE 2
20 FOR i=&c000 TO &ffff
30 POKE i,255
40 NEXT i
```

Vous obtenez sur l'écran une ligne étroite qui est rapidement dessinée vers la droite à partir de l'angle supérieur gauche de l'écran. A la fin de la première ligne, le dessin se poursuit exactement 8 lignes plus bas. Une fois dessinées ces lignes étroites sur toute la surface de l'écran, le dessin reprend d'en haut mais en dessous des lignes déjà dessinées.

Essayez le programme également en mode 1 et en mode 0.

Puis modifiez aussi la ligne 30 ainsi:

```
30 POKE i,1
```

Vous obtenez maintenant une ligne de points qui remplit l'écran verticalement.

Lorsque le programme tourne en mode 2, on voit que les lignes verticales se trouvent sur le côté droit des caractères.

En mode 1, nous obtenons 2 lignes verticales par caractère, en mode 0, 4 lignes.

Nous allons maintenant apporter une dernière modification au programme. Supprimez la ligne 10 du programme et entrez 'MODE 2' en mode direct. L'écran se vide et Ready apparaît dans l'angle supérieur gauche. Appuyez sur la touche de curseur BAS, jusqu'à ce que le message Ready disparaisse de l'écran. Le curseur se trouve maintenant dans la dernière ligne de l'écran. Faites à nouveau tourner le programme.

Le résultat est quelque peu agaçant.

Ce petit programme nous a révélé plusieurs choses importantes d'un seul coup. D'abord nous avons démontré que la mémoire écran commence en &c000

et finit en &FFFF. Curieusement, la taille de la mémoire écran est la même pour les trois modes écran. La seule différence entre les modes réside dans les couleurs.

Cependant on peut se demander à quoi servent 16 K de mémoire écran en mode 0, lorsqu'on ne représente que 20 caractères par ligne, 20 caractères par 25 lignes font 500 caractères sur l'écran. Pourquoi le CPC a-t-il besoin de 16384 cases mémoire pour représenter à l'écran ces 500 caractères?

La réponse est simple. Comme nous l'avons déjà indiqué, le CPC ne possède pas de Ram vidéo dans laquelle chaque caractère serait stocké dans un octet.

En mode 80 colonnes, un caractère sur l'écran occupe 8 octets, en 40 colonnes, un caractère occupe 16 octets et 32 octets en mode 20 colonnes. C'est d'ailleurs ce que montrait le programme qui produisait les lignes verticales.

Le mode 80 colonnes est à cet égard le plus simple à comprendre, puisque chaque bit mis produit un point dans la couleur actuelle d'écriture (pen). Si un bit n'est pas mis, c'est au contraire la couleur du fond de l'écran qui apparaît à cet endroit. Comme en mode 2, il n'y a qu'une couleur d'écriture possible, il n'y a pas d'autres possibilités.

Mais à quoi servent donc en mode 0 les 32 octets nécessaires pour un caractère?

Le fonctionnement des modes 0 et 1 n'est plus aussi simple à expliquer. Nous vous conseillons de taper le petit programme suivant et d'avoir sous les yeux les effets de ce programme, pendant que vous lirez nos explications. Les explications seront alors plus compréhensibles.

```
10 MODE 2
20 REM
30 PRINT "A"
40 FOR adresse=&c000 TO &f800 STEP &800
50 p$=BIN$(PEEK(adresse),8)
60 FOR I=1 TO 8
70 IF MID$(p$,I,1)="1" THEN PRINT "X"; ELSE PRINT ".";
80 NEXT I
```

```
90 PRINT
100 NEXT adresse
```

Faites tourner ce programme et vous obtiendrez une image correspondant à la matrice de 'A'.

Modifiez maintenant l'instruction MODE de la ligne 10 en MODE 1 et faites tourner le programme. Le résultat est assez surprenant. Vous pouviez vous imaginer que seule une moitié de la matrice figurerait dans les octets lus. Mais il semble curieux a priori que la matrice n'utilise qu'une moitié d'octet, soient les bits 4 à 7.

Mais nous nous rapprochons de la solution de cet enigme, lorsque vous modifiez ainsi la ligne 20:

```
20 PEN 2
```

Non seulement la couleur d'écriture (PEN) s'est modifiée, mais la carte bits montrée par notre programme s'est aussi modifiée. Et voilà la solution de notre problème!

Si vous connaissez déjà le CPC, vous savez qu'en mode 40 colonnes, 4 couleurs sont possibles. Ces 4 couleurs sont tout simplement stockées avec le caractère lui-même. En effet 4 bits seulement déterminent les pixels (points de l'écran) allumés et les quartets low et high décident des couleurs (un quartet=un demi-octet, 4 bits). Avec le principe utilisé, le gate array n'a qu'à doubler horizontalement les pixels correspondant à l'affichage, représentant ainsi 8 points, alors que seuls 4 bits sont stockés en mémoire.

En mode 0, pour représenter 20 caractères par ligne, cette méthode est encore étendue. Il n'y a plus ici que deux bits qui contiennent les informations sur les pixels. La position de ces deux pixels à l'intérieur de l'octet détermine la couleur dans laquelle ces pixels doivent être représentés. Il y a ainsi 16 combinaisons possibles, ce qui correspond exactement au nombre de couleurs disponibles. Comme seulement deux pixels sont stockés dans un octet, $4 \times 8 = 32$ octets sont nécessaires pour représenter un caractère avec 16 couleurs différentes possibles.

Essayez à nouveau le programme en mode 0 en utilisant différentes valeurs pour l'instruction PEN. Vous comprendrez vite le fonctionnement.

Les deux premiers points soulevés au début du chapitre sont ainsi éclaircis. Reste cependant le point du 'décalage' de la Ram écran. Ce problème a sa source dans l'électronique du CPC.

Même un Z80 avec une fréquence d'horloge de 4 MHz a besoin d'un certain temps pour décaler un bloc de données de 16 K. Par exemple, pour éviter d'avoir à décaler de 640 cases mémoire, lors du listage d'un programme assez long, la totalité de la zone de Ram vidéo, on a utilisé une propriété du CRTC. Par programmation adéquate des registres 12 et 13 du 6845, l'écran peut commencer pratiquement en n'importe quelle case mémoire paire de la Ram vidéo. Le scrolling (défilement de l'écran) peut ainsi se produire nettement plus vite, puisqu'il suffit de fournir les valeurs adéquates aux registres qui conviennent. La nouvelle ligne dans le bord inférieur de l'écran est vite effacée et remplacée par les nouveaux caractères.

Il n'est pas possible de faire commencer la Ram vidéo à une adresse impaire, par exemple en &C001, du fait de l'utilisation décrite plus haut du signal CCLK comme bit d'adresse.

Le programme suivant montre qu'il est possible de manipuler les registres décrits, même en Basic:

```
10 adrreg = &bc00 : REM registre d'adresse du 6845
20 datreg = &bd00 : REM port du registre de donnée
30 OUT adrreg,13 : REM selectionner le registre
40 FOR offset = 1 TO 40
50 OUT datreg,offset : REM modifier 40 fois
60 for attendre = 1 TO 40 : REM et attendre un peu
70 NEXT attendre,offset
```

Ce programme réalise un scrolling horizontal de l'écran. Sans la boucle de temporisation, le scrolling se déroulerait tellement vite qu'il ne serait pas possible de suivre avec un oeil humain.

Le scrolling vertical peut également être programmé en Basic. Il faut alors modifier les deux registres low-byte et high-byte. Mais comme il s'écoule beaucoup de temps entre les deux instructions OUT, on obtient des phénomènes désagréables à l'écran.

Mais, en ce qui concerne la Ram vidéo, il faut encore relever une

particularité.

Multiplions les valeurs que nous connaissons entre elles.

En mode 2, un caractère se compose de 8 octets. Il y a 80 caractères par ligne et 25 lignes sur l'écran. La place occupée en mémoire est donc de $80 \times 25 \times 8 = 16000$ octets. Mais une zone de 16 K comporte $2^{14} = 16384$ emplacements. Où sont les 384 octets manquants?

Très simple. Ils ne servent à rien, du moins tant qu'il n'y a pas de scrolling de l'écran.

Il est donc possible de placer ici des valeurs à stocker provisoirement. Ces valeurs seront cependant effacées par la première instruction CLS.

Vous vous demandez certainement comment il est donc possible de programmer du graphisme avec une organisation aussi compliquée de la mémoire écran.

Il semble également impossible de lire un caractère à partir de l'écran. Sur d'autres ordinateurs, cela ne pose pas de problème, puisqu'on peut placer un caractère sur l'écran avec POKE et qu'on peut donc lire le contenu de la Ram vidéo avec PEEK.

D'autre part il est normalement assuré que la Ram vidéo commence à une adresse déterminée.

Les choses ne se présentent cependant pas aussi mal que cela peut sembler au premier abord. Le système d'exploitation est en effet en mesure de discerner les adresses de début modifiables ou de déterminer un caractère à partir de la matrice de l'écran, comme cela se produit chaque fois que vous utilisez la touche COPY. Les routines utilisés à cet effet peuvent également être employées dans des programmes en langage-machine que vous aurez réalisés vous-même.

Vous retrouverez bon nombre de ces routines du système d'exploitation dans un prochain chapitre. Nous vous montrons concrètement comment utiliser le graphisme dans un exemple de dessin de rectangles et dans un programme de hardcopy graphique.

1.7 L'interface parallèle 8255

Développé à l'origine par INTEL pour le 8080, le 8255 convient également pour d'autres processeurs comme élément polyvalent d'entrée/sortie. Le 8255 dispose en tout de 24 canaux à travers lesquels les signaux peuvent être sortis ou entrés. Chaque groupe de 8 canaux constitue un port 8 bits et le troisième port peut être scindé en deux moitiés programmables.

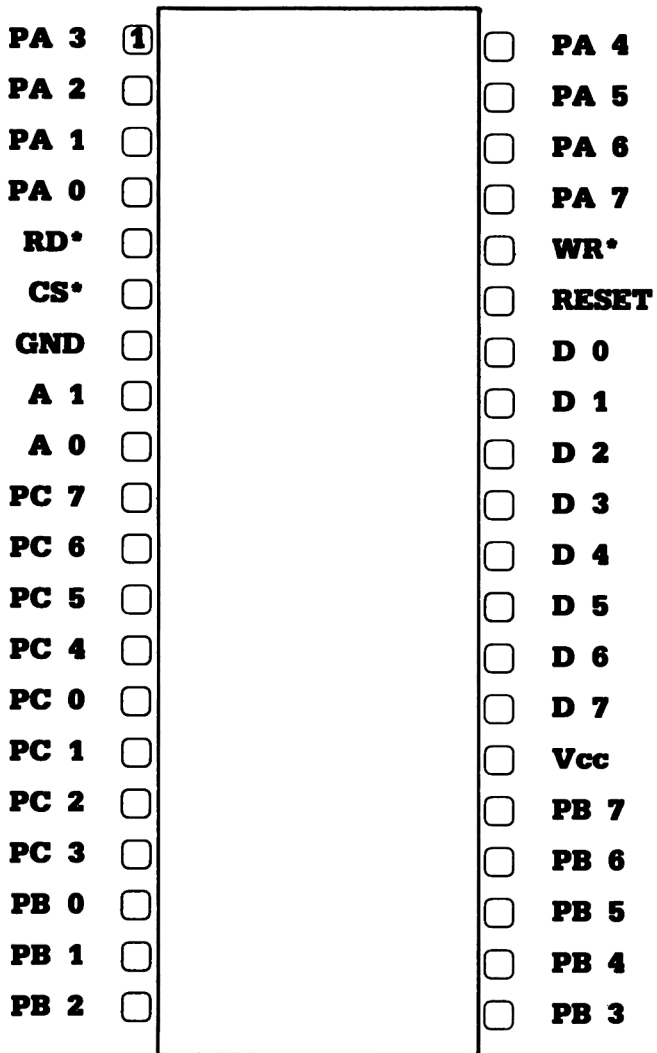
Les principales caractéristiques du 8255 sont:

- 24 connexions d'entrée/sortie programmables
- Alimentation en courant continu de 5 volts
- Entièrement compatible TTL
- Trois puissants modes de travail programmables
- Chaque port programmable séparément
- Courant de sortie de 1 mA pour une tension de 1.5 Volts
- Possibilité de fonction mettre bit/annuler bit

1.7.1 L'affectation des connexions du 8255

L'affectation des pins du 8255 est indiquée par la figure suivante. En voici la signification:

- DO - D7 : Data lines. Ces connexions sont reliées au bus de données du processeur. Elles servent au transfert des données vers et à partir du processeur.
- CS : Chip select. Un low sur ce pôle sélectionne le composant. Les signaux figurant actuellement sur les canaux RD, WR et Data sont acceptés par le 8255.
- RD : Read. Un low sur ce pôle entraîne que le 8255 envoie des données ou des informations d'état au processeur, à travers le bus de données.
- WR : Write devient low lorsque le processeur veut envoyer des données ou des instructions de commande au 8255.



1.7.1.1. Pinout du port parallèle 8255

- AO,A1 : Adress Lines 0 et 1. A travers ces connexions s'opère la sélection entre les trois canaux de données et le registre de commande. Ces connexions sont souvent liées aux deux canaux d'adresse inférieurs du processeur.
- RESET : Un high sur cette entrée rétablit les valeurs initiales de tous les registres, y compris le registre de commande. Les canaux de port sont placés en mode de travail entrée.
- PA0 - PA7 : Port A. Ces huit canaux représentent le port d'entrée/sortie A et peuvent être utilisés au choix en entrée ou en sortie.
- PB0 - PB7 : Port B. Fonctionnement identique au port A
- PC0 - PC7 : Port C. Fonctionnement identique au port A

1.7.2 Les modes de travail du 8255

Avant que nous n'en venions aux quatre registre internes, il nous faut tout d'abord examiner d'un peu plus près les possibilités de ce circuit intégré. Comme nous l'avons indiqué au début, le 8255 dispose de 3 modes de travail possibles:

- Mode de travail 0 : Simple entrée/sortie
- Mode de travail 1 : Entrée/sortie manipulable
- Mode de travail 2 : Bus à deux sens

Le mode de travail 0 est le plus simple et le plus courant. Dans ce mode, il est possible de déterminer si les ports doivent travailler comme canaux d'entrée ou de sortie. Si les canaux sont programmés comme sortie et si le processeur envoie une information sur ces sorties, cette valeur est stockée, et les sorties sont conservées jusqu'à une nouvelle programmation ou jusqu'à un reset.

Les ports programmés comme entrée fournissent lors d'une lecture l'état momentané de ces canaux.

Le sens des données sur le port A aussi bien que sur le port B ne peut être programmé que de façon identique pour tout le port. Il n'est pas

possible d'utiliser par exemple les bits de port PA0, PA3 et PA7 en sortie et les autres bits du même port en entrée.
Le port C peut cependant être programmé en deux moitiés distinctes. Le sens des données de chaque moitié peut être programmé séparément.

Le mode de travail 1 se différencie fondamentalement du mode 0. Dans ce mode de travail, un transfert de données dans un sens est possible avec des signaux hand shake. On ne parle plus alors de trois ports car les deux moitiés du port C sont mises à la disposition des deux autres ports comme signaux de commande et de réception. On parle alors des deux groupes A et B.

Le groupe A comprend le port A et les bits 4 à 7 du port C, le groupe B le port B et les bits 0 à 3 du port C.

Pour programmer facilement le mode 1, il est possible d'utiliser un bit spécial de chaque moitié du port B comme signal d'interruption.

Un tel transfert de données 8 bits est utilisé par exemple sur les interfaces d'imprimante. Un signal indique ici que les données sur les canaux de données sont valables. Un signal rapporté indique si le récepteur, en l'occurrence l'imprimante, est prêt à recevoir des données, ou si les données ont été reçues correctement.

Cette fonction peut être exécutée par le 8255, au choix pour une sortie ou une entrée de données.

Le troisième mode de travail (mode 2) est un mode de travail bidirectionnel. Cette fonction n'est possible qu'avec le port A. Les bits PC3-7 sont utilisés comme signaux de commande et de réception.

Une application possible de ce mode de travail serait la commande d'un lecteur de disquette car les données doivent dans ce cas être transmises aussi bien du lecteur de disquette au processeur que du processeur au lecteur, à travers les mêmes connexions.

Il est d'autre part possible dans les trois modes de travail de mettre ou d'annuler individuellement par instruction les bits programmés en sortie. Les trois modes de travail ainsi décrits peuvent être également combinés. Il est ainsi possible d'utiliser le Port A en mode 0 comme sortie, le port B en mode 1 comme entrée et de programmer les bits restants du port C en entrée.

1.7.3 Commande du 8255, description des registres

Lorsqu'on considère tout d'abord ce nombre troublant de possibilités, on se demande malgré soi comment toutes les possibilités et combinaisons peuvent être programmées avec un seul registre de commande. L'astuce qui rend cela possible est simple. Le bit supérieur du mot de commande est utilisé comme bit témoin. Si ce bit est mis dans le mot de commande, les bits 0 à 6 ont la signification suivante:

Bit 0 : commande la fonction Port C bits 0-3
1=Entrée
0=Sortie

Bit 1 : commande la fonction Port B
1=Entrée
0=Sortie

Bit 2 : sélectionne le mode groupe B
1=Mode de travail 0
0=Mode de travail 1

Bit 3 : commande la fonction Port C bits 4-7
1=Entrée
0=Sortie

Bit 4 : commande la fonction Port A
1=Entrée
0=Sortie

Bit 6,5 : sélectionne le mode groupe A
00=Mode 0
01=Mode 1
1x=Mode 2, bit 5 sans signification

Si par contre le bit supérieur du mot de commande est nul, la fonction 'mettre un bit/annuler un bit' du port C est définie. La signification de ces bits est:

Bit 0 : commande Bit Set/Bit Reset
1=mettre un bit
0=annuler un bit

Bits 3-1: Sélection du bit

000 = PC0

001 = PC1

010 = PC2

011 = PC3

100 = PC4

101 = PC5

110 = PC6

111 = PC7

Les bits 4 à 6 du mot de commande sont sans signification lorsque le bit 7 est nul.

Ce registre de commande ne peut être lu. Il n'est possible que d'écrire. Les registres correspondant aux ports peuvent par contre être lus, même lorsque les ports sont utilisés en sortie. Dans ce cas, la valeur lue correspond à l'état des canaux de port.

L'accès aux quatre registres se fait à travers les pins de connexion A0 et A1. Ces connexions sont décodées dans le 8255 et utilisées comme signaux de sélection de registre. Normalement A0 et A1 du 8255 sont envoyés sur les canaux de même nom du processeur. Il en résulte un adressage transparent sur 4 adresses.

L'affectation aux registres des connexions A0 et A1 est indiquée par le tableau suivant:

A1	A0	
0	0	Registre Port A
0	1	Registre Port B
1	0	Registre Port C
1	1	Registre de commande

1.7.4 L'utilisation du 8255 sur le CPC

Après avoir donné un aperçu des possibilités variées du 8255, nous revenons au fonctionnement pratique de ce composant universel sur le CPC. Comme en fait presque tous les circuits intégrés sur le CPC, le 8255 est également utilisé de façon optimale. Aucun bit n'est inutilisé.

Mais devenons plus concret.

Le 8255 sert le clavier, le chip sonore, le moteur du lecteur de cassette, produit les signaux d'écriture du lecteur de cassette, lit le flux de bits venant du lecteur de cassette, contrôle le signal V Sync du CRT, contrôle si l'imprimante est prête à recevoir, interroge avec un bit l'état du signal EXP du connecteur d'extension, décide à travers un pont si la production de l'image doit se faire suivant la norme PAL ou SECAM en 50 ou 60 Hertz et il reste enfin encore trois bits qui interrogent des ponts lors de la mise sous tension de façon à savoir quel ordinateur vous avez acheté. L'état de ces ponts décide en effet si vous recevrez dans le message d'initialisation, le nom de la firme Amstrad, Awa, Triumph, Schneider ou un autre des 8 noms possibles.

Avoir réalisé toutes ces fonctions avec uniquement les 24 canaux d'entrée/sortie disponibles, témoigne de l'esprit d'économie et de l'inventivité des développeurs de ce matériel.

Le schéma de fonction montre comment le 8255 est connecté. Le bus de données est relié directement au bus de données du processeur. Le signal CS (Chip Select) est produit par le bit d'adresse A11 du processeur. Les pins A0 et A1 du 8255 pour la sélection de registre sont reliés aux pins d'adresse A8 et A9 du processeur.

Comme nous l'avons déjà indiqué, les éléments périphériques du CPC sont appelés à travers des adresses de port. C'est pourquoi le canal RD* du 8255 est relié au signal IORD*.

Ce signal est produit par la combinaison des signaux RD* et IORQ* du Z80 avec une porte logique de l'IC112. Uniquement lorsque IORQ* et RD* sont low, apparaît un low sur le pin 6 de sortie de l'IC 74LS32.

La connexion WR* du 8255 est commandée de même. Ici apparaît un low, venant du pin 3 du 74LS32, lorsqu'aussi bien WR* que IORQ* du Z80 deviennent low sur les pins 1 et 2 de l'IC 112.

Ces données permettent maintenant de déterminer les adresses de port du 8255. Pour, par exemple, écrire une valeur dans le registre 0, le registre de données du port A, les connexions A11, A9 et A8 doivent être low. En écriture binaire, nous obtenons, pour l'octet fort du bus d'adresse, la valeur suivante:

A15	A14	A13	A12	A11	A10	A09	A08
1	1	1	1	0	1	0	0

Ce qui correspond à la valeur hexadécimale &F4.

Les 8 bits d'adresse inférieurs n'interviennent pas dans la sélection du 8255, une valeur entre &00 et &FF est ici possible.

Les bits mis dans l'octet fort ne sont pas non plus nécessaires en réalité à un adressage correct et on pourrait donc avoir l'idée d'utiliser comme octet fort la valeur 00H. Cela marcherait d'ailleurs. Mais comme le décodage des différents circuits intégrés périphériques se produit d'une semblable façon incomplète, les bits doivent être mis, sinon d'autres circuits intégrés tels que le CRTC ou le gate array pourraient se croire également appelés.

Mais revenons à notre exemple. Donc, pour charger une valeur dans le registre A, la valeur &F400 doit être placée sur le bus d'adresse. Ceci peut être obtenu avec les instructions:

```
LD    A,valeur
LD    BC,&F400
OUT   (C),A
```

Le registre de port C peut de même être lu avec les instructions:

```
LD    BC,&F600
IN    A,(C)
```

Les trois ports sont utilisés essentiellement en mode 0. Les 24 canaux d'entrée/sortie sont ainsi disponibles.

Le port A (&F400) est relié aux 8 canaux de données du générateur de son AY-3-8912. Suivant l'action demandée, le port A est programmé comme entrée ou sortie.

S'il est programmé en sortie, les instructions de commande sont envoyées au chip sonore à travers les 8 canaux du port. Vous trouverez le détail de ces instructions de commande dans le chapitre sur la programmation du AY-3-8912. Indiquons simplement pour le moment que le chip sonore dispose également d'un port 8 bits bidirectionnel. Une page de la matrice du clavier est connectée sur ce port. A travers le port A du 8255, il est possible par un détour du port du AY-3-8912 de savoir si une touche est enfoncée. A cet effet, le port A doit bien sûr être programmé en entrée.

Le port B (&F500) est programmé comme port d'entrée. Toutes les interrogations évoquées, hormis celle du clavier, se produisent à travers

ce port. Les différents bits de ce port reçoivent l'affectation suivante:

- Bit 0 : Ce bit interroge l'état du V Sync du CRTC. Comme cette interrogation doit aller très vite, le bit 0 peut être décalé dans le flag carry par simple rotation de la valeur lue avec INP. Il est ainsi possible de connaître rapidement l'état de V Sync.
- Bits 1-3 : Ce bit est relié au pont LK4. Si ce pont est ouvert, le contrôleur vidéo est programmé pour le travail en PAL en 50 Hertz. Un pont fermé entraîne une programmation du CRTC pour la norme SECAM de 60 Hertz pour la fréquence de renouvellement de l'image. Cette possibilité de programmation différente est importante lorsque le CPC doit être utilisé à travers le module MP1 sur un téléviseur.
- Bit 5 : Ce bit interroge l'état du signal EXP du connecteur d'extension.
- Bit 6 : Ce bit restitue l'état d'une imprimante connectée. Comme l'imprimante ne peut pas recevoir de caractères en permanence, il est possible d'interdire un transfert de caractère en fixant cette connexion sur high.
- Bit 7 : Les données fournies par le lecteur de cassette avec un niveau TTL sont lues à travers ce bit. Ici aussi vaut ce que nous disions pour le bit 0. Comme ce canal doit être examiné très rapidement, l'état de ce canal peut être déterminé très vite par une rotation unique du bit 7 vers le flag carry.

Le port C (&F600) est sur le CPC programmé comme port de sortie. Quatre de ses huit canaux lui permettent de commander une partie de l'interrogation du clavier et deux autres bits sont utilisés pour le lecteur de cassette. Les deux bits restants sont employés pour la commande du chip sonore. Comme les canaux du port C peuvent être mis et annulés directement, celui-ci convient particulièrement à ce type de tâches.

Les différents bits sont ainsi utilisés:

Bits 0-3 : Ces bits commandent la matrice du clavier. Les quatre canaux

programmés en sortie sont reliés à l'IC101, un décodeur BCD-décimal.

Ce décodeur met sur la masse une de ses 10 entrées, en fonction de l'information binaire en entrée. Les combinaisons en entrée autorisées sont les valeurs de 0 à 9.

- Bit 4 : Ce bit commande le moteur du lecteur de cassette. Le moteur n'est cependant pas commandé directement, mais à travers un transistor (et un relais commuté à la suite). Si ce bit est sur la masse, le moteur s'arrête. Un high en sortie sur PB4 est conduit par le transistor Q101 et le moteur tourne si la touche PLAY est enfoncée.
- Bit 5 : Les fréquences, qui doivent être reçues par le lecteur de cassette et qui produisent cette si douce mélodie, sont fournies par l'ordinateur à travers ce pin du 8255.
- Bits 6-7 : Ces bits de port sont reliés aux connexions BC1 et BDIR du chip sonore et travaillent comme signaux de chip select et de strobe pour l'AY-3-8912. Vous trouverez une description plus détaillée de ces connexions dans le prochain chapitre sur le générateur de son.

1,8 Le générateur de son programmable AY-3-8912

L'AY-3-8912 de General Instruments est un générateur de son programmable (PSG) de grande classe. Il a été développé pour les Jeux électroniques, afin de doter ceux-ci d'un son particulièrement réaliste alors que les premiers Jeux électroniques ne pouvaient produire que des bruits vraiment monotones. Pour pouvoir être employé le plus universellement possible, le PSG a été doté d'un grand nombre de possibilités d'influencer le son. On pensa en outre lors du développement de ce circuit intégré que, dans pratiquement tous les domaines d'application, il faudrait pouvoir interroger des touches, joysticks ou commutateurs quelconques. C'est pourquoi on a donc également doté ce PSG d'un port parallèle 8 bits.

Les caractéristiques de ce circuit intégré sont les suivantes:

- Trois oscillateurs de son programmables indépendamment
- Un générateur de bruit programmable
- Des sorties analogues entièrement commandées par logiciel
- 15 niveaux de volume étagées par logarithme
- Courbes d'enveloppe programmables
- Compatible TTL
- Alimentation en courant continu de 5 Volts

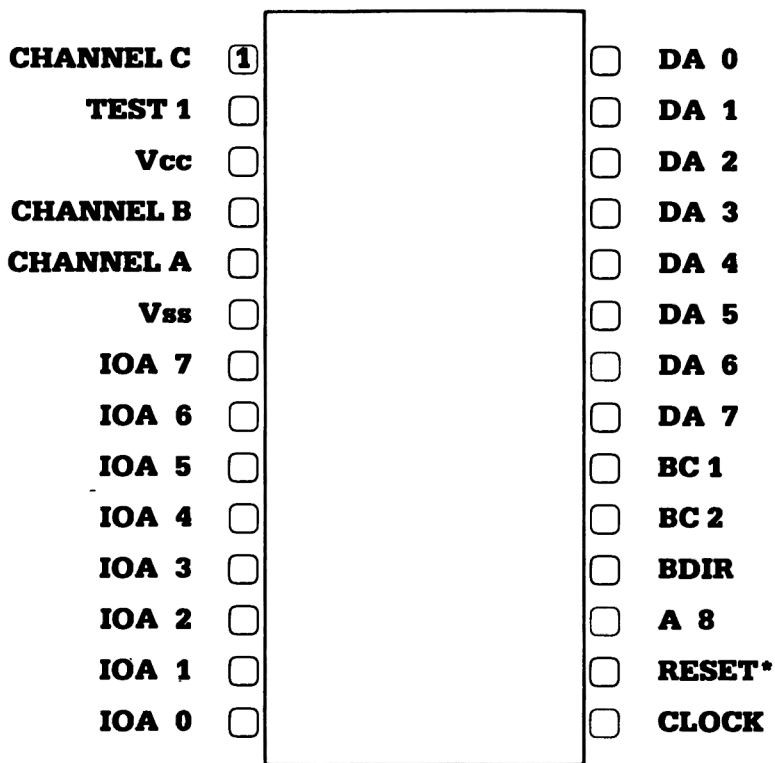
L'AY-3-8912 dispose en tout de 16 registres, dont 15 registres peuvent être utilisés. A travers ces registres peuvent être programmées toutes les possibilités sonores du chip.

Le branchement du PSG peut être divisé en différents blocs de fonction.

Il y a d'abord le bloc des générateurs de son. Les générateurs de son reçoivent un signal d'horloge qui est produit à partir de la division par 16 du signal de l'horloge. Les générateurs de son sont responsables de la production fondamentale des trois fréquences de son carrées.

Le générateur de bruit produit un signal carré en modulation de fréquence dont l'écart de pulsation est influencé par un pseudo générateur de bruit.

Les mixeurs couplent les signaux de sortie des trois générateurs avec le signal de bruit. Le couplage peut être programmé séparément pour chaque



1.8.1.1 Soundchip AY-3-8912

canal.

Le bloc de fonction du contrôle d'amplitude offre deux possibilités à l'utilisateur. D'une part l'amplitude de sortie (le volume) des trois canaux peut être influencée à travers la programmation du registre de volume correspondant.

D'autre part il est possible de les faire influencer de façon variable par le PSG. La sortie du registre de courbe d'enveloppe est alors utilisée pour influencer le volume. Comme la courbe d'enveloppe peut être programmée avec quatre paramètres distincts, les possibilités d'influencer le son sont variées.

Le bloc de fonction du convertisseur D/A est responsable de la production du volume des signaux de sortie. Comme les informations de volume et d'enveloppe sont sous forme de valeurs digitales, elles sont converties dans le convertisseur D/A.

Le dernier bloc de fonction n'a rien à voir avec la production du son. Dans ce bloc sont placés deux ports I/O. Si vous êtes maintenant un peu surpris, c'est que vous nous avez lu attentivement. En effet le chip AY-3-8912 contient deux ports I/O complets dont un seul cependant est branché sur les pins de connexion. Le même chip est utilisé dans l'AY-3-8910, sur lequel les deux ports peuvent être utilisés.

1.8.1 Les connexions du chip sonore

Comme les noms des connexions du PSG ne sont pas suffisamment explicatifs, voici une description détaillée de la fonction des pins:

DAO - 7 : Ces connexions du chip sonore sont reliées au bus de données du processeur. Le nom DA indique que aussi bien des Données que des Adresses (de registre) passent à travers ces connexions.

A8 : Cette connexion peut être comprise comme un signal CHIP-SELECT. Pour appeler des registres du PSG, ce signal doit être high.

B DIR & BC1,2 : La connexion signal B DIR (Bus DIRection) et les connexions BC1 et BC2 (Bus Control) commandent l'accès aux registres sur le PSG. Au premier abord, l'affectation indiquée par le tableau peut paraître curieuse. Mais comme ce circuit intégré fut à l'origine développé comme composant du processeur 1610, un processeur 16 bits spécial de General Instruments, on a pris en compte lors de la conception les propriétés spéciales et des connexions de commande de ce processeur.

B DIR	BC2	BC1	Fonction du PSG
0	0	0	INACTIVE
0	0	1	LATCH ADDRESS
0	1	0	INACTIVE
0	1	1	READ FROM PSG
1	0	0	LATCH ADDRESS
1	0	1	INACTIVE
1	1	0	WRITE TO PSG
1	1	1	LATCH ADDRESS

Dans ce tableau, seules quatre des huit combinaisons ont vraiment un sens. C'est pourquoi la connexion BC2 est souvent mise sur +5 Volts. Le tableau restant n'est donc plus influencé que par les signaux B DIR et BC1 et il se présente ainsi:

B DIR	BC1	Fonction du PSG
0	0	INACTIF, le bus de données du PSG a une valeur en ohm haute
0	1	READ, des données peuvent être lues dans les registres du PSG
1	0	WRITE, des données peuvent être écrites dans le registre du PSG sélectionné
1	1	LATCH, le numéro ou l'adresse du registre du PSG

souhaité est écrit dans le PSG

- ANALOG A : C'est la sortie du canal A. Ici peuvent être retirés les sons produits par le canal A. La tension maximale en sortie est d' 1 Vss.
- ANALOG B : Fonction identique au pin 1, pour le canal B
- ANALOG C : Fonction identique au pin 1, pour le canal C
- IOA7 - 0 : Les connexions IOA représentent le port 8 bits du PSG. Suivant la façon dont elles sont programmées, les connexions travaillent comme sortie ou entrée. Mais on ne peut fixer qu'un même mode de travail pour tout le port. On ne peut avoir simultanément des bits travaillant en entrée et d'autres en sortie.
- CLOCK : De la fréquence de ce signal sont dérivées par division toutes les fréquences de son. La fréquence de ce signal devrait être entre 1 et 2 MHz.
- RESET : Un niveau low sur cette connexion annule les valeurs de tous les registres. Sans reset, les registres contiennent après la mise sous tension des valeurs aléatoires dont la conséquence serait un bruit probablement très peu musical.
- TEST1 : Test1 n'est utilisé que par le constructeur et ne doit pas être connecté en travail normal.
- Vcc : Une tension de +5 Volts est placée sur cette connexion.
- Vss : Ceci est la connexion de masse du PSG.

1.8.2 La fonction des différents registres du 8912

Comme nous avons maintenant vu comment les registres peuvent être appelés fondamentalement à travers les connexions BDIR et BC1, nous allons étudier quelles sont les fonctions remplies par ces registres. Le numéro de registre utilisé dans la liste suivante est identique au numéro qui doit être placé dans le registre d'adresse pour appeler le registre

souhaité.

Il est un fait intéressant qui est que le registre d'adresse conserve son contenu jusqu'à ce qu'il soit à nouveau programmé. On peut donc accéder sans problème plusieurs fois successives à un registre de données, sans devoir chaque fois recharger le registre d'adresse.

Mais voici maintenant la description des registres:

Reg 0,1 : Ces registres déterminent la période et donc la fréquence du signal de son sur ANALOG A. Mais les 16 bits ne sont pas tous utilisés. Tous les 8 bits du registre 0 et les quatre bits inférieurs du registre 1 sont utilisés. La fréquence peut être influencée de façon fine avec le registre 0 ou grossièrement avec le registre 1. Plus la valeur 12 bits de ces registres est petite, plus le son est haut.

Reg 2,3 : Fonction comme Reg 0,1 mais canal B.

Reg 4,5 : Fonction comme Reg 0,1 mais canal C.

Reg 6 : Ce registre influence le générateur de bruit avec ces 5 bits inférieurs.

Reg 7 : Dans ce registre multi-fonctions, les différents bits contrôlent des tâches différentes, comme le montre le tableau suivant:

Bit 0 : mettre/couper le son du canal A 0=mis/1=non

Bit 1 : mettre/couper le son du canal B 0=mis/1=non

Bit 2 : mettre/couper le son du canal C 0=mis/1=non

Bit 3 : mettre/couper le bruit du canal A 0=mis/1=non

Bit 4 : mettre/couper le bruit du canal B 0=mis/1=non

Bit 5 : mettre/couper le bruit du canal C 0=mis/1=non
Bit 6 : Port A comme entrée/sortie 0=entrée/1=sortie
Bit 7 : Port A comme entrée/sortie 0=entrée/1=sortie

- Reg 8 : Ce registre détermine le volume du signal sur le canal A. Les quatre bits inférieurs sont utilisés pour fixer le volume. Le bit 4 a une signification particulière. S'il est mis, le volume est déterminé par le registre de courbe d'enveloppe et le contenu des bits 0 à 3 est alors ignoré.
- Reg 9 : Comme Reg 8 pour le canal B
- Reg 10 : Comme Reg 8 pour le canal C
- Reg 11,12 : Les 16 bits de ces deux registres influencent la période de la courbe d'enveloppe. Le contenu du Reg 11 est considéré comme low byte, c'est-à-dire qu'il influence la période par étapes fines, alors que le Reg 12 est le high byte du générateur de courbe d'enveloppe.
- Reg 13 : Les bits 0 à 3 de ce registre déterminent la forme de la courbe du générateur de courbe d'enveloppe. Il est presque impossible de rendre compréhensible par des mots l'affectation de ces bits. C'est pourquoi les courbes d'enveloppe sont montrées dans le graphique 1.8.2.1.

1.8.3 Le fonctionnement de l'AY-3-8912 sur le CPC

Nous allons nous intéresser dans cette section à la connection concrète et certaines choses plus concrètes pour l'utilisation du chip sonore sur le CPC. Comme la description des registres qui précède était nécessairement abstraite et peut-être pas très aisément compréhensible, vous comprendrez mieux, après avoir lu ce chapitre, certaines particularités du PSG.

Jetons d'abord un coup d'oeil sur le schéma de fonction.

Le PSG y figure comme IC 102.

Les pins 3, 17 et 19 sont sur +5 Volts. L'AY-3-8912 reçoit son alimentation électrique à travers le pin 3. Comme BC2 (pin 19) et A8 (pin 17) sont sur +5 Volts, ils n'interviennent pas dans la sélection des registres.

Les connexions de commande des registres restantes BC1 (pin 20) et BDIR (pin 18) sont reliées aux bits de port PC6 et PC7 du 8255. Suivant l'état de ces connexions, des adresses de registre peuvent être communiquées au PSG ou des données peuvent être écrites ou lues dans le PSG.

Le transfert d'adresse et de données proprement dit se produit à travers les connexions D0 à D7 du PSG qui sont reliées au port A du 8255. Suivant l'action demandée, le port A doit être programmé comme entrée ou sortie.

Le signal de l'horloge sur le pin 15 est un signal carré d'une fréquence de 1 MHz. Ce signal est fourni par le gate array par division de la fréquence quartz. De ce signal sont dérivées par division de fréquence toutes les fréquences de son et de courbe d'enveloppe.

Le port I/O du PSG est relié au clavier et à la connexion pour le joystick. Vous trouverez dans un prochain chapitre une description détaillée du clavier et du joystick, nous ne nous intéressons ici qu'aux possibilités sonores du chip sonore.

Les connexions les plus importantes de ce circuit intégré sont certainement les trois sorties analogues A, B et sur les pins 1, 4 et 5. Ces sorties fonctionnent comme ce qu'on appelle des sorties Open-Emitter. Pour pouvoir sortir une tension alternative du son, des résistances sont nécessaires qui commutent entre sortie et masse. C'est la fonction des

résistances R121, R122 et R123.

Le signal sonore est mixé par ces trois résistances à travers les trois résistances R114, R115 et R116 et il se présente alors sous forme d'un signal mono sur la connexion 1 du port d'extension. Ce signal mono est cependant également conduit sur la prise CP001. De là, ce signal arrive à l'amplificateur et au haut-parleur internes.

Les trois sorties sont cependant en outre conduites également vers la prise stéréo à l'arrière de l'ordinateur. A cet effet, le signal du canal B est envoyé de façon identique sur les deux canaux stéréo, à travers les résistances R118/R119. Les sorties A et C sont chacune envoyées directement sur un des canaux stéréo, à travers un condensateur de découplage (R177 et R120).

Ce type de branchement rend même possible, avec une habile programmation, d'obtenir de véritables effets stéréo. Il serait par exemple imaginable de ne sortir d'abord un son que sur le canal A. Au bout de quelque temps, le même son pourrait être sorti en plus sur le canal B. On pourrait, ce faisant, faire monter lentement le volume du signal sur le canal B, alors que le volume du signal serait par contre réduit de façon symétrique. Le résultat serait qu'il semblerait que le son se promène d'un coin de la pièce vers le milieu entre les deux baffles. De là, il peut si nécessaire continuer vers l'autre coin.

Ces effets sont mêmes possibles en Basic avec la puissante instruction sound. Le manuel d'utilisation comporte cependant des contradictions dans l'indication de la répartition des trois canaux de son sur les deux canaux stéréo. Observez-le après avoir relié votre CPC à une chaîne stéréo. Seuls les sons du canal B apparaissent sur les deux canaux de la chaîne stéréo.

Mais comment le PSG produit-il au fond les sons? Examinons un peu comment les choses se produisent en détail sur un canal.

Comme nous l'avons déjà indiqué, tous les sons sont dérivés du signal de l'horloge sur le pin 15. Le signal d'horloge est d'abord divisé par 16. Il en résulte sur le CPC une fréquence de commande et 62,5 KHz. Cette fréquence est alors conduite vers un diviseur de fréquence programmable. Suivant le contenu des registres du générateur de son, la fréquence de commande est ou non à nouveau divisée, pour obtenir la fréquence voulue. Les développeurs de ce circuit intégré ont à cet égard fait montre de beaucoup d'astuce. La chaîne de division n'est pas seulement constituée de flip-flops qui peuvent diviser la fréquence par deux. Par une technique de branchement spéciale, des facteurs impairs de division sont

également possibles. La fréquence de commande peut tout-à-fait être divisée par 3 ou par 17. C'est uniquement ainsi que toutes les valeurs nécessaires peuvent être produites dans la zone de fréquences élevées.

Si vous consultez l'annexe du manuel du CPC, vous trouvez pour la note Ré de la quatrième octave une valeur de période de 27. Comment cette valeur est-elle obtenue?

La première fois que nous nous sommes posé cette question, nous nous sommes arraché les cheveux. Quels qu'aient été les calculs que nous faisons, nous n'obtenions pas de valeur raisonnable. Ce n'est que plusieurs heures et plusieurs litres de café plus tard que l'idée nous vint que le magnifique tableau fournit dans le manuel du CPC devait être faux. L'entrée de la période dans l'instruction SOUND produit une fréquence qui se situe exactement une octave en dessous de celle indiquée. L'entrée de 'SOUND 1,284,100' ne produit pas la fréquence attendue de 440 Hertz mais exactement 220 Hertz!

La formule correcte pour le calcul de la période est donc:

$$\text{PERIODE}=\text{ROUND}(62500/\text{FREQUENCE})$$

Le tableau a été vraisemblablement réalisé en partant d'une fréquence de commande de 2 MHz.

Mais considérons encore la production des sons sur le PSG. Le contenu des registres du générateur de son détermine donc le facteur de division pour le signal sonore. Si le registre 0 du PSG reçoit la valeur 100, le registre 1 la valeur 0, la fréquence de commande sera divisée par 100. Sur la sortie de la chaîne de division du canal A se trouve un signal d'une fréquence de 625 Hertz.

Ce signal ne peut cependant pas encore être retiré sur la sortie A. Il faut d'abord que le canal correspondant soit activé. Ceci est obtenu en annulant le bit correspondant du registre 7. Comme nous avons choisi dans notre exemple le canal A, nous devons annuler le bit 0. Mais il faut, ce faisant, considérer l'état des autres bits. Sur le CPC, cela signifie concrètement qu'il ne faut pas modifier le bit 6 involontairement car sinon le clavier est bloqué.

Mais pour le moment on ne peut entendre encore aucun son, parce que le volume de chaque canal doit être fixé. Pour le canal A, c'est le registre 8 qui est responsable. Une valeur de 1 ne produit qu'un son très doux, alors qu'une valeur de 15 donne le volume maximal.

Si nous mettons le bit 4 de registre de volume, les informations contenues dans les bits 0 à 3 seront ignorées. Ce sont maintenant les registres 11, 12 et 13 qui déterminent le volume. Le volume n'est plus alors fixé sur une valeur mais variable.

Considérons d'abord le registre 13. Ce registre porte le nom officiel de 'ENVELOPE SHAPE/CYCLE CONTROL REGISTER'. Sa fonction sera illustrée plus aisément grâce à un exemple.

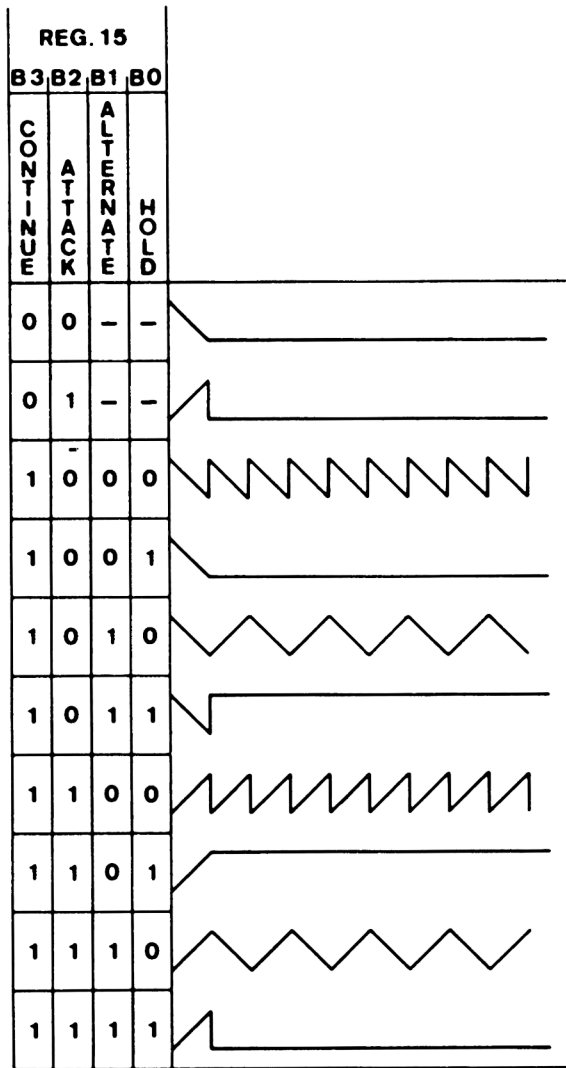
Après que nous ayons fourni les valeurs adéquates aux registres 0, 1, 7 et 8, écrivons donc dans le registre 13 la valeur 12. Les bits 2 et 3 sont maintenant mis, alors que les 2 bits inférieurs sont annulés.

Le tableau fourni dans la description des registres montre pour cette combinaison une suite de dents montant lentement et retombant rapidement. En pratique, cela signifie que le volume du son monte tout d'abord lentement jusqu'au maximum. Puis le son est coupé et le volume recommence à monter. Cet état demeure jusqu'à ce qu'une nouvelle instruction soit envoyée au registre 13.

La durée de la montée du volume peut être fixée à travers les registres 11 et 12. Ces registres influencent de façon analogue aux registres des générateurs de son une autre chaîne de division programmable sur le PSG. La chaîne de division reçoit un signal qui correspond au signal de l'horloge divisé par 256. Cela donne une fréquence de 3906,25 Hertz correspondant à une période d'environ 250 microsecondes.

Si une valeur 1 est écrite dans le registre 11 et une valeur 0 dans le registre 12 qui travaille comme high-byte, le volume du son est réellement conduit en 250 microsecondes de 0 jusqu'au volume maximum. Cela figure cependant déjà dans la zone des sons audibles et produit un sifflement net qui est superposé au son véritablement souhaité. C'est pour cette raison que les valeurs de registre choisies sont toujours nettement plus élevées. Avec la valeur maximale (255 dans Reg 11 et Reg 12), la montée jusqu'au volume maximum dure 16,8 secondes.

L'altération du volume à travers le registre d'enveloppe n'est pas utilisée par le logiciel du CPC. L'instruction ENV influence le volume du son uniquement à travers des manipulations des autres bits inférieurs du registre de volume. L'instruction ENT du CPC n'a pas d'équivalent sur le PSG. Cette fonction est produite par une modification habile des registres du générateur de son.



1.8.2.1 Courbes d'enveloppe du PSG

1.9 Les interfaces du CPC

Le concept d'interface peut être défini comme un point de liaison entre l'ordinateur et le monde extérieur. Le monde extérieur peut être aussi bien un autre ordinateur qu'une imprimante ou un autre périphérique, qu'un appareil de mesure ou un homme. D'après cette définition du monde extérieur, nous ne décrirons pas seulement dans ce chapitre les connexions figurant à l'arrière de l'ordinateur mais également le clavier, la connexion du moniteur et le lecteur de cassette.

Les interfaces les plus importantes pour l'utilisateur sont le clavier et le moniteur car celles-ci représentent le contact immédiat avec l'ordinateur. Commençons donc par ces deux interfaces.

1.9.1 Le clavier

Le clavier du CPC comprend en tout 74 touches. Comme les deux touches SHIFT sont branchées parallèlement, il y donc 73 touches différentes qui peuvent être interrogées.

La matrice dans laquelle les touches sont rangées comprend 8 fois 10 canaux. Comme les joysticks peuvent également être interrogés à travers cette matrice, 79 positions de touche sont donc occupées en tout. Le second joystick connecté directement sur le premier n'est pas connecté à des positions autonomes de la matrice, les branchements correspondants sont parallèles à des touches du clavier.

Du point de vue électronique, le clavier est interrogé à travers le 8255 et le chip sonore. Cela fonctionne à peu près de la façon suivante.

Le 8255 fournit aux sorties de port PC0 à PC3 une moitié d'octet qui est transformée par le décodeur IC101 en une information décimale. Suivant l'information figurant en entrée, une des dix sorties devient low. Ce décodeur, un 74LS145 est pour cette raison également appelé décodeur BCD-décimal. Si l'information en entrée n'est pas comprise entre 0 et 9, toutes les sorties du décodeur sont sur high.

Le port parallèle du chip sonore est programmé pour l'interrogation du clavier comme port d'entrée. Si aucun signal ne se trouve sur ces entrées, on obtient lors de la lecture du port un 1 sur toutes les entrées, en tout donc &FF.

Soit maintenant une information en entrée sur le décodeur de &04. La

sortie pin 5 deviendra donc low. Mais le chip sonore ne le prendra pas en compte tant qu'aucune touche correspondante ne sera enfoncée. Le fait d'appuyer sur la touche ESC n'aura par exemple aucun effet à ce moment puisque la sortie pin 8 du décodeur est high. Mais si par contre la touche ESPACE est enfoncée, la valeur fournie par le chip sonore se modifiera. A cause de la touche enfoncée, le bit 7 du port est maintenant sur la masse et nous obtenons du chip sonore la valeur &7F.

Toutes les touches sont examinées 50 fois par seconde. A cet effet, les valeurs 0 à 9 sont sorties l'une après l'autre sur les quatre sorties utilisées du port C et la valeur du chip sonore est examinée après chaque sortie. Si des touches enfoncées sont alors enregistrées, les touches enfoncées sont placées dans un tableau et sont si nécessaire converties en numéros de touche et en caractères correspondants.

Un fait très pratique sur le clavier est que jusqu'à 20 caractères sont stockés provisoirement. Dans des programmes Basic, on peut déjà commencer à faire des entrées alors que l'ordinateur n'a pas terminé certains calculs ou qu'il est occupé à la sortie sur écran. L'interrogation du clavier n'est bloquée que lors de l'utilisation du lecteur de cassette car il ne reste pas assez de temps pour cela, étant donné le timing très précis de ces opérations. La seule exception est la touche ESC qui est en effet nécessaire pour permettre une éventuelle interruption d'une opération avec le lecteur de cassette.

Le clavier a par ailleurs une petite particularité. Essayez par exemple d'appuyer simultanément sur les touches J, K et L. De façon très surprenante, vous voyez apparaître en outre un H sur l'écran. Cela se produit toujours lorsque vous appuyez sur trois touches qui constituent les angles d'un carré dans la matrice du clavier, de même par exemple que 123 ou DFG. Dans ce cas apparaît simultanément le quatrième caractère de la matrice.

Ce 'défaut' est sans grande conséquence et vous pouvez par ailleurs également interrompre des programmes en appuyant simultanément sur les touches 2, 3 et E.

1.9.2 La connexion vidéo

La connexion vidéo du CPC fournit tous les signaux nécessaires au fonctionnement d'un moniteur. Il est à cet égard indifférent qu'il

1.9.3 Le lecteur de cassette

La cassette est un moyen de stockage de données standard remarquable pour un prix très intéressant.

Même si vous possédez déjà ou acquerrez plus tard un lecteur de disquette, le lecteur de disquette continuera certainement à vous rendre de bons services. Comme les disquette utilisées par le CPC sont tout de même relativement chères, la cassette peut être utilisée comme un moyen bon marché d'effectuer des copies.

Le lecteur de cassette lui-même est un type de vente courante, ce qui explique la présence de la touche PAUSE qui est en fait parfaitement inutile.

L'électronique du lecteur a toutefois été adaptée aux besoins du CPC. Le signal de sortie est un signal carré avec une amplitude d'environ 5 Volts. Il peut ainsi être traité directement par le bit 7 du port B du 8255.

L'amplificateur audio qui permet d'entendre le son du CPC a été également placé sur la platine du lecteur.

Mais venons-en au format d'écriture.

Le lecteur de cassette ne peut fondamentalement stocker les données que bit par bit. Chaque octet à stocker doit donc être décomposé en ses différents bits et être transmis sous cette forme. Cette décomposition est réalisée par le processeur par logiciel, le bit supérieur étant à cet effet envoyé en premier au lecteur de cassette.

Le signal fourni par le 8255 pour le lecteur de cassette est un signal carré. Chaque bit est marqué par une vibration carrée, dans laquelle la phase low est exactement aussi longue que la phase high. On dit également que le signal carré a un rapport de 1:1. Un bit 0 nécessite moitié moins de temps qu'un bit 1.

C'est pourquoi les indications sur la vitesse d'écriture ne peuvent être que des indications imprécises. Il est évident qu'un bloc composé uniquement d'octets 0 sera sauvegardé en deux fois plus de temps qu'un bloc d'à peu près la même taille ne comportant que des &FF. Mais comme la répartition des bits 0 et 1 dans un bloc de données est à peu près égale, on peut s'en tenir aux indications de 1000 baud (1 baud=1 bit par

seconde) pour SUPER-SAVE et de 2000 baud pour SPEED-LOAD.

Chaque fichier, qu'il s'agisse d'un fichier programme ou d'un fichier de données, peut comporter au maximum 65536 octets. Les fichiers sont écrits par blocs comportant chacun au maximum 2048 octets. Chaque bloc comprend au maximum huit segments de données de 256 octets. Devant chaque bloc est écrit un header, c'est-à-dire une tête de bloc

Bien qu'il n'y ait pas de liaison électrique avec l'amplificateur et le haut-parleur, il est possible, même si le volume est baissé, de suivre le chargement et la sauvegarde de données et de programmes.

Le header de bloc est facile à identifier à l'oreille. On entend en effet un long ton égal suivi de quelques octets qu'il n'est toutefois pas possible de distinguer à l'oreille.

Le ton long et égal est une série de 2048 bits 1. Après ces bits vient un seul bit 0 puis un octet de synchronisation. La longue suite de bits 1 est nécessaire à l'ordinateur pour déterminer la vitesse (baud-rate). Le bit 0 indique à l'ordinateur que cette tête est terminée et l'octet sync est nécessaire pour distinguer entre l'information du header et les données.

L'information du header figure dans une zone de données longue de 64 octets qui est transmise devant chaque bloc de 2K de données. Dans ce fichier header figurent les informations sur le fichier lui-même, par exemple le nom, si le fichier est ou non protégé, s'il s'agit d'un programme Basic ou d'un fichier Ascii et quelle est la longueur du programme.

Octets 0- 15 : Nom du fichier, si moins de 16 octets, rempli avec 00

Octet 16 : Numéro de bloc, dans cet octet figure le numéro qui sera affiché lors du chargement ou également avec Catalog.

Octet 17 : Si dans cet octet figure une autre valeur que 00, il s'agit du dernier bloc du fichier.

Octet 18 : Cet octet contient le type de fichier. L'information est codée dans les différents bits. La signification des bits vient à la suite de ce tableau.

Octets 19,20 : Ces octets contiennent la longueur des informations du fichier. Si le bloc, donc les 2 K, est entièrement écrit, ces octets contiennent la valeur &0800. Dans le dernier ou unique bloc, figure ici le nombre d'octets du bloc.

Octets 21,22 : Ces octets indiquent l'adresse de chargement, à partir de laquelle les données ont été écrites à l'origine. Pour les programmes Basic, c'est l'adresse décimale 368, pour les fichiers binaires, donc pour le langage-machine, c'est normalement l'adresse où tourne le programme en mémoire.

Octet 23 : Si le contenu de cet octet est différent de 0, il s'agit du premier bloc du fichier.

Octets 24,25 : Ces octets contiennent la longueur du fichier.

Octets 26,27 : Les possibilités de ces octets ne sont malheureusement pas soutenues directement par le Basic du CPC. Elles contiennent l'adresse de début d'un fichier en langage-machine, qui n'est pas en effet nécessairement identique à l'adresse de chargement. Ces octets permettent par programmation de réaliser un 'auto-start'.

Les octets restants 28 à 63 du header ne sont pas utilisés par le système d'exploitation et sont à la disposition des programmeurs chevronnés.

Mais voici maintenant le codage des bits de l'octet 18 du header:

Bit 0 : Si ce bit est mis, le fichier correspondant est déclaré protégé. Les programmes protégés peuvent être également produits en Basic avec 'SAVE "NOM",p'.

Bits 1-3 : Ces bits déterminent le type de fichier. Bien que trois bits permettent 8 différents types de fichier, seuls les types de fichier programme Basic (0), fichier binaire (1) et fichier de données ascil (3) sont utilisés.

Bits 4-7 : Ces bits comportent normalement un 0, seuls les fichiers Ascil ont un 1 dans le bit 4.

Comme nous l'avons déjà indiqué, les informations stockées dans les différents blocs sont encore subdivisées en différents segments. Chaque segment se compose de 256 octets de données et d'octets de checksum (contrôle du total). La checksum de chaque segment est calculée d'après une formule spéciale et permet de vérifier lors de la lecture du fichier si les bits ont été correctement transmis. Dès lors que la checksum calculée ne correspond pas aux valeurs lues, le READ ERROR B est affiché. Le READ ERROR A indique qu'un bit a été lu dont la durée était trop longue par rapport aux valeurs calculées pour les bits nuls ou 1. Cette erreur se produit souvent lors de la lecture de programmes, lorsque la cassette qui coinceait lors de la sauvegarde est maintenant fluide. La troisième erreur possible est le READ ERROR D. Cette erreur ne devrait se produire que rarement car elle signale que le bloc lu est plus long que les 2048 octets autorisés. Cela ne peut toutefois se produire que si l'utilisateur écrit dans les informations du header, lors de la sauvegarde, des valeurs plus grandes que celles autorisées.

Vous connaissez certainement l'instruction Basic 'SPEED WRITE par'. Suivant les paramètres utilisés, les données sont stockées sur la cassette à une vitesse moyenne de 1000 ou 2000 baud. Ceci n'atteint cependant pas encore la vitesse la plus grande possible. Par l'utilisation d'une routine du système d'exploitation, la vitesse (baudrate) peut être fixée à toute valeur comprise entre 700 et environ 3600 baud. La routine nécessaire est à l'adresse &BC68. Elle attend des paramètres dans deux registres et fixe la vitesse d'écriture en fonction de ces paramètres. Une valeur est transmise à la paire de registres HL qui détermine la vitesse (baudrate). La formule pour déterminer cette valeur est:

Baudrate=333333/moitié de la longueur d'un bit nul

Cela donne pour 1000 baud une vitesse de 666 microsecondes pour un bit nul; un bit 1 dure exactement le double.

L'électronique utilisée dans le lecteur de cassette a cependant une particularité. Si des bits nuls et des bits 1 sont lus tour à tour, l'électronique essaye de combler les différences de durée. Les bits 1 deviennent de ce fait plus courts, alors que les bits nuls apparaissent comme des impulsions plus longues qu'on ne l'aurait attendu après l'écriture. Pour cette raison, une compensation anticipée doit être

exécutée et les bits nuls sont écrits plus brièvement, alors que les bits 1 sont écrits avec des durées légèrement plus longues. Ces durées nécessaires pour la compensation anticipée sont transmises à la routine dans l'accumulateur.

Pour des tentatives de fixer la vitesse d'écriture la plus rapide, qui est à moitié fiable, il suffit de transmettre dans l'accumulateur une valeur de 10. Pour écrire avec une vitesse de 3600 baud, il faut activer la routine suivante:

```
LD HL,93
LD A,10
CALL &BC68
RET
```

Ces quelques octets peuvent facilement être placés dans la mémoire avec les lignes suivantes:

```
10 MEMORY HIMEM - 10
20 FOR I = 1 TO 9
30 READ X : POKE HIMEM + I,X
40 NEXT I
50 CALL HIMEM + 1
60 DATA &21,&5D,&00,&3E,&0A,&CD,&68,&BC,&C9
```

Ne craignez pas de faire varier quelque peu les valeurs dans HL et dans l'accumulateur (les deuxième et cinquième valeurs de la ligne de Data), pour déterminer la plus haute fréquence d'écriture possible. Elle dépend des cassettes utilisées. Mais les propriétés de rotation régulière de votre lecteur de cassette jouent également un rôle non négligeable.

Si les valeurs sélectionnées sont trop petites, le CPC ne peut plus alors tenir les durées réclamées et vous obtenez comme résultat le message d'erreur WRITE ERROR A.

Encore un conseil pour finir:

Vous avez certainement remarqué que lorsque vous sauvegardez de très longs programmes avec de nombreuses variables, cela peut durer jusqu'à 15 minutes jusqu'à ce que les données ou le programme soient sauvegardées. Cela vient du fait que le CPC nécessite pour la sauvegarde une zone de 2K

pour les blocs à transférer. Ce buffer est placé dans la limite supérieure de la mémoire. Si cette zone est toutefois occupée par des variables, ces variables sont recopiées dans une autre zone de la mémoire. Ce procédé est comparable à la redoutable garbage collection qui se produit toujours lorsqu'il n'y a plus de place suffisante en mémoire pour les chaînes de caractères et les tableaux.

Le délai d'attente provoqué par le transfert des variables peut cependant être notablement réduit si ce buffer de 2K est déjà installé et protégé au début de chaque programme. Un début de programme possible pourrait se présenter ainsi:

```
10 OPENOUT "DUMMY"  
20 MEMORY HIMEM - 1  
30 CLOSEOUT  
40  
50 'RESTE DU PROGRAMME
```

Ce procédé n'a bien sûr de sens que si vous travaillez dans le programme en question avec des fichiers. Si ce n'est pas le cas, vous pouvez renoncer à ces lignes de programme et entrer simplement l'instruction CLEAR avant la sauvegarde. Toutes les variables définies auparavant seront ainsi supprimées et l'installation du buffer de cassette se fera sans délai notable.

1.9.4 L'interface d'imprimante centronics

On trouve sur tout ordinateur quelque chose qu'on considère comme pouvant être amélioré. Sur le CPC, c'est sans conteste l'interface imprimante. On a malheureusement trop économisé ici.

Nous ne pensons pas à la réalisation mécanique de la connexion. Le choix fait par le constructeur est certainement le moins cher pour lui mais il n'est pas non plus sans avantage pour le possesseur de l'ordinateur car les câbles de connexion nécessaires sont bon marché et vraiment fiables. La cause de notre 'mauvaise humeur' est le fait que l'interface ne dispose que de 7 bits. La plus part des imprimantes, y compris celle proposée par AMSTRAD pour le CPC, ont une entrée 8 bits et donc de nombreuses commandes et possibilités de ces imprimantes ne peuvent être obtenues que par des détours, ou même ne peuvent pas être obtenues du tout.

Mais considérons d'abord la structure électronique de cette interface. L'interface se compose principalement de l'IC106, latch 8-pôles 74LS273. Les huit différents latches travaillent comme des flip-flops, l'information envoyée sur les entrées est stockée avec une bascule high-low sur l'entrée d'horloge pin 11 et elle est disponible sur les sorties, jusqu'à un RESET ou à une nouvelle programmation, quelles que soient les modifications sur les signaux d'entrée.

Le signal d'horloge dont la bascule high-low déclenche le stockage des valeurs d'entrée est produit avec la porte logique OR 74LS32, IC112, pins 11, 12 et 13. La sortie pin 11 devient low, lorsque les deux entrées sont low.

La connexion de l'imprimante est également appelée à travers l'adressage de port. C'est pourquoi le signal IOWR* se trouve sur une entrée de la porte logique OR et que le canal d'adresse A12 se trouve sur l'autre entrée.

Comme sur les autres éléments périphériques, le décodage est ici donc également très incomplet. Les canaux d'adresse qui ne sont pas utilisés pour le décodage doivent donc être high pour éviter des collisions avec d'autres adresses de port utilisées. Ceci donne une adresse de port effective de &EFxx.

Les entrées du latch de l'imprimante sont reliées au bus de données du processeur. Les sorties se trouvent sur la connexion de l'imprimante. Seul le bit 7 est envoyé au port Centronics à travers une porte logique NAND de l'IC110 utilisée comme inverseur. Ce bit représente le signal strobe nécessité par l'imprimante. Ce signal est normalement high. Mais si l'ordinateur veut envoyer un caractère à l'imprimante, il envoie l'octet à transmettre sur les canaux de données et place peu après le signal strobe sur low. L'octet à transmettre est ainsi accepté par l'imprimante.

A condition toutefois que le signal busy de l'imprimante soit low. L'état du signal busy est interrogé par le bit 6 du port B du 8255.

Mais comment le signal strobe peut-il être produit? Rien de plus simple. Chaque octet à transmettre est d'abord ANDé avec &7F. Le bit supérieur de l'octet est ainsi supprimé de façon certaine. Cet octet est sorti sur le port de l'imprimante par une instruction OUT.

Les bits à transmettre se trouvent maintenant déjà sur l'imprimante, mais le signal strobe est toujours high, à travers l'inverseur. C'est pourquoi on met ensuite avec OR &80 le bit 7 de la valeur à sortir qui est

également sortie sur le port imprimante. La valeur à transmettre n'a pas été modifiée, seul le signal strobe est devenu low à travers l'inverseur. Ce signal doit cependant redevenir également high, c'est pourquoi le bit supérieur est à nouveau supprimé avec AND et l'octet est à nouveau sorti. Un octet a été ainsi envoyé de l'ordinateur à l'imprimante.

La sortie sur l'imprimante ne pose pas de problème en Basic. Mais même en langage-machine, il n'est pas nécessaire d'écrire soi-même toute cette procédure. Il y a plusieurs routines système qui vous évitent une bonne part de ce travail de programmation.

Il y a d'abord la routine dont l'entrée est en &BD2B. A travers cette routine, vous pouvez sortir un caractère sur l'imprimante. Le caractère doit chaque fois se trouver dans l'accumulateur. Cette routine teste en outre si l'imprimante est 'busy'. Si l'imprimante ne répond pas dans un délai de 0.4 secondes, la routine revient avec un flag carry nul. Il faut alors faire une nouvelle tentative avec le même caractère. Cette routine est également utilisée par l'interpréteur Basic. Si la transmission est réussie, le carry est mis. Le prochain caractère peut alors être envoyé.

Une autre routine a son entrée trois octets plus loin (&BD2E). Cette routine peut être utilisée pour examiner l'état de l'imprimante. Si aucune imprimante n'est connectée ou si l'imprimante répond 'busy', si elle ne peut donc pas recevoir de caractères pour le moment, cette routine revient avec un carry mis, sinon le carry est supprimé.

La troisième routine exploitable (&BD31) accomplit toutes les procédures nécessaires à la sortie d'un caractère sur l'imprimante. Le programmeur doit cependant tester alors auparavant si l'imprimante est prête à recevoir puis transmettre le caractère voulu dans l'accumulateur. Si le test de l'état de l'imprimante est négligé, le caractère peut éventuellement se perdre dans le 'vide'.

Comment ces routines peuvent être mises en oeuvre, nous vous l'indiquerons plus tard dans cet ouvrage. Nous vous montrerons en effet pour l'exemple d'un hardcopy de texte et de graphisme, comment utiliser ces routines et d'autres.

Mais il convient de tenir compte d'une autre particularité de cette connexion Centronics.

La disposition des contacts du port d'imprimante incite à se procurer les

fiches nécessaires ainsi qu'un bout de câble plat pour réaliser soi-même un tel câble. Si les connecteurs sont en outre des pinces crocodile, même des possesseurs de CPC peu doués manuellement peuvent réaliser un tel câble en 5 à 10 minutes. Toutes les imprimantes Centronics peuvent être alors utilisées.

Mais lors du premier essai de fonctionnement, vous aurez une grosse surprise. L'imprimante dépense curieusement le papier très généreusement. Une ligne vide est ajoutée après chaque ligne imprimée.

La raison en est la suivante:

Le CPC ajoute à la fin de chaque ligne imprimée la suite de caractères CR/LF (Carriage Return, Line Feed) c'est-à-dire la suite d'instructions pour retour de chariot et passage à la ligne. Le papier avance donc d'une ligne. De plus, et sans raison très claire, le pin 14 de la connexion centronics du CPC est cependant encore relié à la masse. Cela produit sur la plus part des imprimantes un passage à la ligne supplémentaire, de sorte qu'une ligne vide est ainsi toujours produite.

La solution est dans ce cas l'interruption du canal menant au pin 14. Après avoir écarté ce canal et éventuellement installé des commutateurs dans l'imprimante si nécessaire comme par exemple sur Epson, tout devrait fonctionner correctement.

1.9.5 La connexion du joystick

La connexion du joystick est certainement utilisée principalement dans un but qui ustifie son nom: comme entrée pour l'interrogation d'un joystick. A travers 7 des 9 connexions disponibles, il est cependant également possible d'interroger d'autres touches ou commutateurs. Par programmation et en renonçant aux interruptions et à l'interrogation du clavier, ces sept connexions pourraient même être employées comme sortie. Les connexions de joystick sont en effet reliées au port bi-directionnel du chip sonore et pourraient travailler comme sortie, sous les contraintes indiquées. Le port Centronics est cependant plus facile à manipuler pour effectuer une sortie.

Comme nous l'avons déjà décrit au chapitre 1.9.1, les joysticks sont considérés comme des touches du clavier. C'est pour cette raison que les 7 entrées nécessaires du port du chip sonore sont placées sur la prise du joystick. Deux sorties du décodeur BCD-décimal IC101 sont encore en outre

placées sur la prise.

Tous les cinquantièmes de seconde, le clavier est interrogé entièrement. L'état des joysticks est également interrogé à cette occasion. Pour les programmeurs Basic, l'état des joysticks est fourni par la fonction JOY(numéro). L'état des joysticks pourrait être également déterminé simplement avec INKEY. Mais également pour les fans de l'assembleur, il est possible de déterminer facilement l'état des joysticks. La routine système &BB24 fournit dans le registre double HL l'état actuel des joysticks. En appelant cette routine, on obtient l'état du joystick 0 dans le registre H et le registre L vaut pour le joystick 1. Le codage des touches joystick suit le même schéma qu'avec la fonction JOY(x).

1.9.6 Le connecteur d'extension

Cette interface est la plus universelle du CPC. Sur cette carte de conducteurs à 50 pôles se trouvent, outre tous les signaux du processeur, différents signaux de commande. C'est ici que sont connectées toutes les extensions du système.

La signification des signaux 3 à 39 nous est connue puisqu'elle découle de la description du processeur. C'est pourquoi nous allons nous limiter ici aux connexions restantes.

Sur le pin 1 figure encore une fois le signal sonore. Ce signal n'est toutefois que mono, les trois canaux sont conduits ici.

Les pin 2 et 49 sont reliés à la masse de l'alimentation électrique.

Une particularité est constituée par le signal BUS-RESET* sur le pin 40. En plaçant ce signal à low, on provoque un reset du système. Malheureusement, le CPC vide toute la mémoire lors d'un reset. Ce signal n'est donc comme signal d'alarme pas plus efficace que le fait de couper puis de rallumer l'ordinateur.

Sur le pin 41 figure le signal reset proprement dit pour les extensions extérieures. Notez cependant que tous les composants ne peuvent pas être alimentés avec ce signal. Le 8255 a par exemple besoin de ce signal sous sa forme inversée.

Les deux signaux ROMEN* et ROMDIS sont très intéressants. Le signal ROMEN* qui se trouve sur le pin 42 signale par son niveau low un accès à la Rom intégrée de 32 K. Cet accès peut cependant être interdit par un niveau high sur le pin 43, ROMDIS. La totalité de la Rom intégrée peut donc être ainsi remplacée par des Roms ou Eproms extérieures. Par un décodage approprié des canaux d'adresse, il est cependant également possible de ne masquer et remplacer que des zones déterminées de la Rom intégrée.

Les deux signaux RAMRD* und RAMDIS ont une fonction semblable pour les accès en lecture sur la Ram interne. Ces signaux sur les pins 43 et 44 peuvent être utilisés pour échanger par exemple des zones de mémoire déterminées avec des Roms ou même des Rams.

La commande de Rams extérieures n'est cependant pas très simple sur le CPC. La principale difficulté vient du fait que le signal WR* pour les Rams internes n'est pas produit par le processeur mais par le Gate Array. Cette impulsion d'écriture ne peut malheureusement (à notre connaissance) être empêchée par aucune astuce de programmation, de sorte qu'un accès en écriture à une Ram externe adresse toujours également et écrit sur la Ram interne.

Le signal CURSOR envoyé sur le pin 46 est fourni avec une programmation appropriée par le contrôleur vidéo. Le CRTC dispose en effet de la possibilité offerte par le curseur électronique. Suivant la programmation, un signal carré d'une fréquence d'environ 1.5 ou 3 Hertz apparaît sur cette sortie. Mais il est également possible de programmer sur cette connexion des niveaux low ou high permanents.

Après l'allumage du CPC, c'est un niveau low permanent qui figure ici.

L'entrée LPEN (Light Pen) sur le pin 47 est reliée directement avec l'entrée light-pen du CRTC. Ce circuit intégré dispose de tous les registres nécessaires pour la gestion du lightpen.

L'utilisation du light pen, surtout en graphisme haute résolution est cependant difficilement réalisable sur le CPC car le contrôleur vidéo fournit certes l'adresse MA de la position actuelle du light-pen mais il n'indique pas l'adresse RA actuelle. Du fait de la structure spéciale de la Ram vidéo, cette indication est cependant nécessaire si l'on veut dessiner sur l'écran avec le light-pen.

L'entrée pin 48 porte la désignation EXP* et est reliée au port B du 8255

Bit 4. Une extension extérieure peut placer cette connexion sur la masse et se faire ainsi remarquer par le système d'exploitation.

Le dernier signal à évoquer, sur le Pin 50, est le signal d'horloge du processeur. Ce signal avec une fréquence de 4 MHz est par exemple utilisé par le contrôleur du lecteur de disquette.

2 LE SYSTEME D'EXPLOITATION

Derrière ce nom qui ne dit rien au non initié, se cache le coeur de l'ordinateur. C'est ici qu'est réalisée la liaison entre programme de l'utilisateur et le matériel.

L'interpréteur Basic doit à cet égard être considéré lui-même comme un programme qui accède à travers le système d'exploitation à l'électronique de l'ordinateur.

La structure du système d'exploitation est organisée logiquement et clairement en sections ou packs dont chacune a une fonction particulière. Cela commence au niveau inférieur par le MACHINE PACK qui est la partie la plus proche de l'électronique et qui sert par exemple le port d'imprimante, les registres de son, etc..., cela continue avec le SCREEN PACK qui contrôle l'écran et qui est appelé par le TEXT PACK ou le GRAPHICS PACK.

Un examen plus approfondi montre que chaque pack est strictement délimité et fermé et que la communication avec les autres packs ne se fait qu'à travers certaines interfaces bien définies. En outre, chaque pack dispose d'une zone de Ram propre qu'il emploie comme mémoire de travail. L'appel des routines se produit en règle générale à travers des vecteurs de la Ram ou, plus rarement, directement à travers l'adresse de la Rom.

Cela incline à supposer que le système d'exploitation, probablement à cause de peu de temps disponible, a été écrit par plusieurs programmeurs, chacun étant responsable d'un ou plusieurs packs et après qu'on se soit entendu uniquement sur les interfaces.

Quoi qu'il en soit, cette structure claire et l'accès par des vecteurs à tous les coins et recoins ouvrent au programmeur des horizons insoupçonnés et tout à fait inconnus jusqu'ici.

Citons simplement comme exemple la possibilité d'écrire une routine pour une véritable imprimante 8 bits (sans parler du problème de la connexion) et de rendre cette routine utilisable par le système simplement en modifiant le vecteur MC WAIT PRINTER.

Cette indication doit également vous servir d'avertissement: ne craignez pas d'utiliser les routines du système d'exploitation, mais ne les utilisez qu'à travers les vecteurs! Il se pourrait en effet que quelqu'un d'autre (cartouche Rom) ait déplacé quelques vecteurs pour faire exécuter certaines fonctions par des routines propres.

Vous constaterez à l'usage qu'il est possible d'écrire des programmes

propres en un minimum de temps, pour peu qu'on utilise scrupuleusement les vecteurs. Ce qui est entièrement nouveau, c'est que même les routines arithmétiques du Basic tournent avec ce mécanisme ce qui peut vous permettre d'une part d'y faire exécuter vos propres calculs et d'autre part d'y placer vos propres programmes si vous souhaitez par exemple une plus grande précision.

Puisque nous vous avons montré notre enthousiasme pour les vecteurs, c'est aussi avec eux que nous commencerons dans le chapitre suivant.

2.1 Les vecteurs du système d'exploitation

Nous vous présentons dans les pages suivantes les adresses de la Ram à travers lesquelles vous pouvez appeler des routines du système d'exploitation ou que vous pouvez au besoin modifier pour faire exécuter certaines fonctions par vos propres programmes.

La fonction de la routine est indiquée en quelques mots lorsque le nom même de la routine n'est pas suffisamment explicite. Vous trouverez des indications plus précises sur certaines parties dans les introductions des différents 'packs'.

Il s'agit pour une part de routines complètes qui ont été copiées ici et au beau milieu desquelles il vous est possible de sauter en cas de besoin et pour une autre part de RST 1 ou RST 5 suivie de l'adresse Inline (voyez à ce sujet le chapitre 1.1.2) qui concerne la Rom.

Vous pouvez lire dans l'annexe 4.1 où ces routines figurent dans la Rom.

B900 KL U ROM ENABLE
B903 KL U ROM DISABLE
B906 KL L ROM ENABLE
B909 KL L ROM DISABLE
B90C KL ROM RESTORE Réactive l'ancienne configuration Rom
B90F KL ROM SELECT Active une Rom d'extension (théoriquement, il peut y en avoir Jusqu'à 252)
B912 KL CURR SELECTION Quelle Rom d'extension est actuellement en fonction ?
B915 KL PROBE ROM De quel type d'extension de la Rom s'agit-il?
B918 KL ROM DESELECT Reconstituer extension de la Rom précédente
B91B KL LDIR
B91E KL LDDR
B921 KL POLL SYNCHRONOUS Y a-t-il un Event avec une priorité supérieure à

celle de l'Event actuel?

B939 RST 7 INTERRUPT ENTRY CONT'D

B97C KL LOW PCHL CONT'D

B982 RST 1 LOW JUMP CONT'D

B9A8 Préparer configuration et exécuter saut

B9B1 KL FAR PCHL CONT'D

B9B9 KL FAR ICALL CONT'D

B9BF RST 3 LOW FAR CALL CONT'D

BA10 KL SIDE PCHL CONT'D

BA16 RST 2 LOW SIDE CALL CONT'D

BA2E RST 5 FIRM JUMP CONT'D

BA4A KL L ROM ENABLE CONT'D

BA54 KL L ROM DISABLE CONT'D

BA5E KL U ROM ENABLE CONT'D

BA72 KL ROM RESTORE CONT'D

BA7E KL ROM SELECT CONT'D

BA83 KL PROBE ROM CONT'D

BA8C KL ROM DESELECT CONT'D

BAA2 KL CURR SELECTION CONT'D

BAA6 KL LDIR CONT'D

BAAC KL LDDR CONT'D

BACB RST 4 RAM LAM CONT'D

BADC RAM LAM (IX) correspond à ld a,(ix)

BB00 KM INITIALISE

BB03 KM RESET

BB06 KM WAIT CHAR Attendre un caractère du clavier

BB09 KM READ CHAR Aller chercher un caractère du clavier s'il y en a un

BB0C KM CHAR RETURN placer caractère dans buffer clavier pour prochain accès

BB0F KM SET EXPAND Constituer chaîne d'extension

BB12 KM GET EXPAND Retirer caractère de chaîne d'extension

BB15 KM EXP BUFFER Affecter mémoire pour chaîne d'extension

BB18 KM WAIT KEY Attendre frappe d'une touche

BB1B KM READ KEY Aller chercher numéro de touche, si une touche a été frappée

BB1E KM TEST KEY Une touche a été frappée?

BB21 KM GET STATE Aller chercher état SHIFT

BB24 KM GET JOYSTICK

BB27 KM SET TRANSLATE Recevoir entrée dans table clavier (premier niveau)

BB2A KM GET TRANSLATE Aller chercher entrée dans table du clavier (premier niveau)
BB2D KM SET SHIFT Comme BB27 pour le deuxième niveau
BB30 KM GET SHIFT Comme BB2A pour le deuxième niveau
BB33 KM SET CONTROL Comme BB27 pour le troisième niveau
BB36 KM GET CONTROL Comme BB2A pour le troisième niveau
BB39 KM SET REPEAT Fixer fonction de répétition pour touche déterminée
BB3C KM GET REPEAT La fonction de répétition d'une touche déterminée est-elle activée
BB3F KM SET DELAY Fixer fréquence et vitesse de répétition des touches
BB42 KM GET DELAY Aller chercher paramètres ci-dessus
BB45 KM ARM BREAK Autoriser touche Break
BB48 KM DISARM BREAK Verrouiller touche Break
BB4B KM BREAK EVENT Exécuter routines si touche Break frappée
BB4E TXT INITIALISE
BB51 TXT RESET
BB54 TXT VDU ENABLE Des caractères peuvent être écrits sur l'écran
BB57 TXT VDU DISABLE Interdire représentation des caractères
BB5A TXT OUTPUT Représenter ou exécuter caractère (de commande)
BB5D TXT WR CHAR Représenter caractère
BB60 TXT RD CHAR Lire caractère de l'écran
BB63 TXT SET GRAPHIC Activer ou désactiver la représentation de caractères de commande
BB66 TXT WIN ENABLE Fixer dimensions fenêtre texte actuelle
BB69 TXT GET WINDOW Quelles dimensions a la fenêtre actuelle
BB6C TXT CLEAR WINDOW Vider fenêtre de texte actuelle
BB6F TXT SET COLUMN
BB72 TXT SET ROW
BB75 TXT SET CURSOR
BB78 TXT GET CURSOR
BB7B TXT CUR ENABLE Autoriser le curseur (programme utilisateur)
BB7E TXT CUR DISABLE Verrouiller le curseur (utilisateur)
BB81 TXT CUR ON Autoriser le curseur (système d'exploitation)
BB84 TXT CUR OFF Verrouiller le curseur
BB87 TXT VALIDATE Curseur dans la fenêtre texte?
BB8A TXT PLACE CURSOR Activer curseur
BB8D REMOVE CURSOR Désactiver curseur
BB90 TXT SET PEN Fixer couleur du premier plan
BB93 TXT GET PEN Couleur du premier plan?
BB96 TXT SET PAPER Fixer couleur fond

BB99 TXT GET PAPER Couleur fond?
BB9C TXT INVERSE Echanger entre elles les couleurs actuelles du fond et du premier plan
BB9F TXT SET BACK Activer/désactiver mode transparent
BBA2 TXT GET BACK Mode transparent?
BBA5 TXT GET MATRIX Aller chercher adresse de la carte points d'un caractère
BBA8 TXT SET MATRIX Fixer l'adresse de la carte points (définie par l'utilisateur) d'un caractère déterminé
BBAB TXT SET M TABLE Fixer adresse de départ et premier caractère d'une matrice de points définie par l'utilisateur
BBAE TXT GET M TABLE Adresse de départ et premier caractère d'une matrice utilisateur
BBB1 TXT GET CONTROLS Aller chercher adresse de la table de saut des caractères de commande
BBB4 TXT STR SELECT Choisir fenêtre de texte
BBB7 TXT SWAP STREAMS Les paramètres (couleurs, limites des fenêtres, etc...) de deux fenêtres de texte sont échangés entre eux
BBBA GRA INITIALISE
BBBD GRA RESET
BBC0 GRA MOVE ABSOLUTE
BBC3 GRA MOVE RELATIVE
BBC6 GRA ASK CURSOR Où est le curseur actuel?
BBC9 GRA SET ORIGIN
BBCC GRA GET ORIGIN
BBCF GRA WIN WIDTH Fixer limites gauche et droite de la fenêtre graphique
BBD2 GRA WIN HEIGHT Fixer limites supérieure et inférieure de la fenêtre graphique
BBD5 GRA GET W WIDTH Limites gauche et droite de la fenêtre graphique?
BBD8 GRA GET W HEIGHT Limites supérieure et inférieure de la fenêtre graphique
BBDB GRA CLEAR WINDOW Supprimer fenêtre graphique
BBDE GRA SET PEN Fixer couleur d'écriture
BBE1 GRA GET PEN Couleur d'écriture?
BBE4 GRA SET PAPER Fixer couleur fond
BBE7 GRA GET PAPER Couleur fond?
BBEA GRA PLOT ABSOLUTE Fixer point graphique (absolu)
BBED GRA PLOT RELATIVE Fixer point graphique (relativement au curseur actuel)

BBF0 GRA TEST ABSOLUTE Point mis? (absolu)
 BBF3 GRA TEST RELATIVE Point mis (relativement au curseur actuel)
 BBF6 GRA LINE ABSOLUTE Tracer ligne de position actuelle à position absolue.
 BBF9 GRA LINE RELATIVE Tracer ligne de position actuelle à distance relative
 BBFC GRA WR CHAR Ecrire un caractère dans la position graphique actuelle
 BBFF SCR INITIALISE
 BC02 SCR RESET
 BC05 SCR SET OFFSET Fixer adresse de départ du premier caractère relativement à l'adresse de base de la Ram vidéo
 BC08 SCR SET BASE Fixer adresse de base de la Ram vidéo
 BC0B SCR GET LOCATION Début actuel de l'écran? (Base+offset)
 BC0E SCR SET MODE
 BC11 SCR GET MODE
 BC14 SCR CLEAR Vider l'écran
 BC17 SCR CHAR LIMITS Aller chercher nombres maxi de lignes et de colonnes de l'écran (suivant le mode)
 BC1A SCR CHAR POSITION
 BC1D SCR DOT POSITION
 BC20 SCR NEXT BYTE Augmenter une adresse d'écran donnée d'une position de caractère.
 BC23 SCR PREV BYTE Diminuer l'adresse d'écran d'une position.
 BC26 SCR NEXT LINE AUGMENTER L'adresse d'écran d'une ligne.
 BC29 SCR PREV LINE Diminuer l'adresse d'écran d'une ligne.
 BC2C SCR INK ENCODE
 BC2F SCR INK DECODE
 BC32 SCR SET INK Affecter couleur(s) à une Ink-#.
 BC35 SCR GET INK Couleur(s) à une Ink-#?
 BC38 SCR SET BORDER Composer couleur(s) du cadre.
 BC3B SCR GET BORDER Couleur(s) du cadre?
 BC3E SCR SET FLASHING Fixer périodes de clignotement.
 BC41 SCR GET FLASHING Périodes de clignotement?
 BC44 SCR FILL BOX Remplir fenêtre existante avec une couleur (positions relatives aux caractères, suivant le mode).
 BC47 SCR FLOOD BOX Remplir fenêtre existante avec une couleur (positions sont adresses d'écran, indépendantes du mode).
 BC4A SCR CHAR INVERT Pour un caractère inverser couleur de premier plan et couleur du fond.
 BC4D SCR HW ROLL Décaler l'écran d'une ligne vers le haut ou d'une ligne

vers le bas (selon le hardware).
BC50 SCR SW ROLL Décaler l'écran d'une ligne vers le haut ou d'une ligne vers le bas (selon le software).
BC53 SCR UNPACK Agrandir matrice de caractère (pour mode0/1).
BC56 SCR REPACK Refondre matrice de caractère dans sa forme originale.
BC59 SCR ACCESS Fixer caractère de commande visible/invisible.
BC5C SCR PIXELS Fixer point à l'écran.
BC5F SCR HORIZONTAL Tracer ligne horizontale.
BC62 SCR VERTICAL Tracer ligne verticale.
BC65 CAS INITIALISE
BC68 CAS SET SPEED Fixer vitesse d'écriture.
BC6B CAS NOISY Entrée/sortie de messages de cassette.
BC6E CAS START MOTOR
BC71 CAS STOP MOTOR
BC74 CAS RESTORE MOTOR Rétablir ancienne position de moteur.
BC77 CAS IN OPEN
BC7A CAS IN CLOSE
BC7D CAS IN ABANDON Fermer aussitôt fichier d'entrée.
BC80 CAS IN CHAR Lire caractère (du buffer).
BC83 CAS IN DIRECT Entrer tout le fichier dans la mémoire.
BC86 CAS RETURN Rentrer le caractère lu le dernier dans le buffer.
BC89 CAS TEST EOF Fin de fichier?
BC8C CAS OUT OPEN
BC8F CAS OUT CLOSE
BC92 CAS OUT ABANDON Fermer aussitôt fichier de sortie.
BC95 CAS OUT CHAR Ecrire caractère (dans le buffer).
BC98 CAS OUT DIRECT Ecrire zone de mémoire définie sur cassette (sans passer par le buffer).
BC9B CAS CATALOG
BC9E CAS WRITE Ecrire bloc.
BCA1 CAS READ Lire bloc.
BCA4 CAS CHECK Comparer bloc sur bande avec contenu de la mémoire.
BCA7 SOUND RESET
BCAA SOUND QUEUE Placer le son à la queue.
BCAD SOUND CHECK Encore de la place dans la queue?
BCB0 SOUND ARM EVENT Block d'événement pour provoquer la libération d'une place dans la queue.
BCB3 SOUND RELEASE Permettre des sons.
BCB6 SOUND HOLD Tenir aussitôt les sons
BCB9 SOUND CONTINUE Continuer de traiter les sons auparavant tenus.

BCBC SOUND AMPL ENVELOPE Dresser la courbe d'enveloppe de volume.
 BCBF SOUND TONE ENVELOPE Dresser la courbe d'enveloppe de son.
 BCC2 SOUND A ADDRESS Prendre l'adresse d'une courbe d'enveloppe de volume.
 BCC5 SOUND T ADDRESS Prendre l'adresse d'une courbe d'enveloppe de son.
 BCC8 KL CHOKE OFF Ramener kernel en arrière.
 BCCB KL ROM WALK Quelles extensions- rom?
 BCCE KL INIT BACK Ajouter extensions-rom.
 BCD1 KL LOG EXT Ajouter extension résidente.
 BCD4 KL FIND COMMAND Chercher instruction dans tous les domaines ajoutés de mémoire.
 BCD7 KL NEW FRAME FLY Installer et suspendre bloc d'événement.
 BCDA KL ADD FRAME FLY Suspendre bloc d'événement.
 BCDD KL DEL FRAME FLY Sortir bloc d'événement.
 BCE0 KL NEW FAST TICKER Comme BCD7.
 BCE3 KL ADD FAST TICKER Comme BCDA.
 BCE6 KL DEL FAST TICKER Comme BCDD.
 BCE9 KL ADD TICKER Installer et suspendre bloc ticker.
 BCEC KL DEL TICKER Sortir bloc ticker.
 BCEF KL INIT EVENT Installer bloc d'événement.
 BCF2 KL EVENT Expulser le bloc d'événement.
 BCF5 KL SYNC RESET Effacer Sync Pending Queue.
 BCF8 KL DEL SYNCHRONOUS Effacer un certain bloc de la pending queue.
 BCFB KL NEXT SYNC Suivant SVP.
 BCFE KL DO SYNC Exécuter routine d'événement.
 BD01 KL DONE SYNC Routine d'événement prête.
 BD04 KL EVENT DISABLE
 BD07 KL EVENT ENABLE
 BDOA KL DISARM EVENT Fermer bloc d'événement(compteur négatif).
 BD0D KL TIME PLEASE
 BD10 KL TIME SET
 BD13 MC BOOT PROGRAM Ramène le système d'exploitation en arrière et transmet la commande à une routine dans (h1).
 BD16 MC START PROGRAM
 BD19 MC WAIT FLYBACK Attendre le retour du rayon.
 BD1C MC SET MODE
 BD1F MC SCREEN OFFSET
 BD22 MC CLEAR INKS
 BD25 MC SET INKS
 BD28 MC RESET PRINTER

BD2B MC PRINT CHAR Imprimer caractère si possible.
BD2E MC BUSY PRINTER Imprimante encore en fonction?
BD31 MC SEND PRINTER Imprimer caractère (attendre que cela marche).
BD34 MC SOUND REGISLTER Fournir des données au Sound Controller.
BD37 JUMP RESTORE Initialiser tous les vecteurs de saut.

Les vecteurs suivants sont utilisés en BASIC.

BD3A EDIT
BD3D FLO Copier variable de (de)=>(hl)
BD40 FLO Int=>Flo
BD43 FLO valeur 4 octets =>Flo
BD46 FLO Flo=>Int
BD49 FLO Flo=>Int
BD4C FLO FIX
BD4F FLO INDT
BD52 FLO
BD55 FLO Chiffre multiplié par 10^a .
BD58 FLO Addition
BD5B FLO Soustraction
BD5E FLO Soustraction
BD61 FLO Multiplication
BD64 FLO Division
BD67 FLO Chiffre multiplié par 2^a
BD6A FLO Comparaison
BD6D FLO Modification du caractère initial
BD70 FLO SGN
BD73 FLO DEG/RAD
BD76 FLO PI
BD79 FLO SQR
BD7C FLO Elévation à la puissance
BD7F FLO LOG
BD82 FLO LOG10
BD85 FLO EXP
BD88 FLO COS
BD8E FLO TAN
BD91 FLO ATN
BD94 FLO valeur 4 octets *256=>Flo
BD97 FLO RNDIn1
BD9A FLO Set RND Seed

BD9D FLO RND
BDA0 FLO Prendre dernière valeur-RND.
BDA3 INT
BDA6 INT
BDA9 INT Recevoir signe initial en b.
BDAC INT Addition
BDAF INT Soustraction
BDB2 INT Soustraction
BDB5 INT Multiplication avec signe
BDB8 INT Division avec signe
BDBB INT MOD
BDBE INT Multiplication sans signe
BDC1 INT Division sans signe
BDC4 INT Comparaison
BDC7 INT Changement de signe
BDCA INT SGN

Ici commencent ce qu'on appelle les indirections. Ce sont des sauts dans le système d'exploitation qui ne sont pas affectés globalement mais individuellement par chaque pack, lorsque son RESET ou son INITIALISE est exécuté.

BDCD TXT DRAW CURSOR Curseur sur l'écran
BDD0 TXT UNDRAW CURSOR Curseur éteint
BDD3 TXT WRITE CHAR Caractère sur l'écran
BDD6 TXT UNWRITE Lire caractère de l'écran
BDD9 TXT OUT ACTION Représenter ou exécuter caractère
BDDC GRA PLOT fixer un point
BDDF GRA TEST point ?
BDE2 GRA LINE Tracer une ligne
BDE5 SCR READ Aller chercher point dans l'écran
BDE8 SCR MODE CLEAR vider écran avec Ink#0
BDEE KM TEST BREAK Touche Break enfoncée ?
BDF1 MC WAIT PRINTER Envoyer caractère à l'imprimante

2.2 La Ram du système d'exploitation

Vous trouverez ici une liste du système d'exploitation de la Ram, pour autant que nous ayons réussi à découvrir la signification des différentes adresses.

Vous ne devez cependant entreprendre de manipulation directe de ces adresses que si vous savez auparavant quels effets peuvent résulter de ces manipulations. Vous pouvez constater en effet que toutes les fonctions importantes du système d'exploitation viennent fureter par ici, y compris des choses aussi considérables par exemple que la table de saut du TEXT SCREEN.

Nous comprenons bien sûr, car c'est pour cela que vous avez acheté cet ouvrage, que vous ayez envie de faire des testes. Donc, allez-y! Mais n'oubliez pas de sauvegarder auparavant le programme qui se trouve en mémoire, car il pourrait pâtir de vos essais.

B08B Pointeur de pile Basic
B08D pointeur début des chaînes de caractères
B08F Pointeur fin des chaînes
B09A Pointeur de pile du stringdescriptor
B09C Pile du stringdescr.
B0BA Stringdescriptor
B0C1 Type de variable
B0C2 INTvar / AdrFLOvar / PointSTRdesc
B100 KL Start Int Pending Queue
B104 KL div. flags pour rout. int.
B105 KL sp save
B187 KL Timer low
B189 KL Timer high
B18B KL Timerflag
B18C KL Start Frame Fly Chain
B18E KL Start Fast Ticker Chain
B190 KL Start Ticker Chain
B192 KL Count for Ticker
B193àKL Start Sync Pending Queue
B195 KL Priorité évènement courant
B196 KL Instruction à exécuter
B1A8 KL Rom d'extension actuelle
B1A9 KL Entrée Rom actuelle
B1AB KL Configuration de Rom actuelle

B1C8 SCR curr. Screen Mode
B1CA SCR Adr. Screen Start
B1CB SCR High Byte Screen Start
B1CC SCR Write Indirection
B1CF SCR Configuration bits suivant le mode
B1D7 SCR Flash Periods
B1D8 SCR Flash Period 1ère couleur
B1D9 SCR Mémoire de couleur 2ème couleur
B1DA SCR Mémoire de couleur 2ème couleur
B1EA SCR Mémoire de couleur première couleur
B1FB SCR Flag jeu de couleur actuel
B1FD SCR curr. Flash Period
B1FE SCR Event Block: Set Inks
B20C TXT fenêtre d'écran actuelle
B20D TXT Start Params Fenêtre 0
B21C TXT Params Fenêtre 1
B22B TXT Params Fenêtre 2
B23A TXT Params Fenêtre 3
B249 TXT Params Fenêtre 4
B258 TXT Params Fenêtre 5
B267 TXT Params Fenêtre 6
B276 TXT Params Fenêtre 7
B285 TXT Position actuelle du curseur (ligne,col)
B287 TXT Flag fenêtre (0=écran entier)
B288 TXT Fenêtre actuelle haut
B289 TXT Fenêtre actuelle gauche
B28A TXT Fenêtre actuelle bas
B28B TXT Fenêtre actuelle droite
B28C TXT Roll Count actuel
B28D TXT act. Cursor Flag
B28E TXT VDU Flag (0=disabled)
B28F TXT Pen actuel
B290 TXT Paper actuel
B291 TXT Background Mode actuel
B293 TXT Graph Char Write Mode (0=disabl)
B294 TXT 1er caractère matrice utilisateur
B296 TXT Adr. User Matrix
B2B8 TXT Compteur de caractères Control Buffer
B2B9 TXT Start Control Buffer
B2C3 TXT Table de saut caractères de contrôle

B328 GRA X Origin
B32A GRA Y Origin
B32C GRA actuelle coord. X
B32E GRA actuelle coord. Y
B330 GRA coord X Fenêtre GRA gauche
B332 GRA coord X Fenêtre GRA droite
B334 GRA coord Y Fenêtre GRA haut
B336 GRA coord Y Fenêtre GRA bas
B338 GRA Pen
B339 GRA Paper
B342 GRA Buffer de calcul coord X
B344 GRA Buffer de calcul coord Y
B4DE KM Exp. String Pointer
B4E0 KM Put Back Buffer
B4E1 KM Adr. Start Exp Buffer
B4E3 KM Adr. Fin Exp Buffer
B4E5 KM Adr. Start Exp Buffer libre
B4E7 KM Shift Lock State
B4E8 KM Caps Lock State
B4E9 KM Delay
B4EB KM Key State Map
B4ED KM Key 16...23
B4F1 KM Joystick 1
B4F4KM Joystick 0
B4F5 KM pendant scanning touches enfoncées
B4FF KM Multihit contr. à B4F5
B50D KM Break Event Block
B541 KM Adr. Key Translation Table
B543 KM Adr. Key SHIFT Table
B545 KM Adr. Key CTRL Table
B547 KM Adr. de la table de répétition
B551 SOUND ancienne act. sound (après HOLD)
B552 SOUND actuelle Activité sound
B555 SOUND Sound Event Block
B55C SOUND Params canal A
B59B SOUND Params canal B
B5DA SOUND Params canal C
B60A SOUND courbes d'enveloppe de volume
B6FA SOUND courbes d'enveloppe de ton
B800 CAS Cass. Message Flag

B802 CAS Input Buffer Status
B803 CAS Adr. Start Input Buffer
B805 CAS Pointer Input Buffer
B807 CAS File Header Input
B847 CAS Output Buffer Status
B848 CAS Adr. Start Output Buffer
B84A CAS Pointer Output Buffer
B84C CAS File Header Output
B8D1 CAS Cass. Speed
B8DD EDIT Insert Flag

2.3 Utilisation des routines du système d'exploitation

Le CPC contient plusieurs centaines de routines ou fonctions dont certaines sont très utiles et parfaitement utilisables par les programmeurs. On trouve par exemple de telles routines pour l'interrogation du clavier, pour sortir un caractère sur l'écran, pour gérer les fenêtres ou pour commander l'imprimante.

Malgré la masse de fonctions dont dispose le système d'exploitation, il y a cependant des choses que le CPC ne sait pas faire de lui-même. C'est ainsi qu'il manque par exemple la possibilité de sortir le contenu de l'écran, texte ou graphisme sur une imprimante connectée au système.

Cette possibilité appelée 'Hardcopy', nous allons vous la montrer dans deux exemples. Dans le premier exemple il s'agira d'un hardcopy de texte uniquement, qui fonctionne avec n'importe quelle imprimante connectée. La seconde routine de hardcopy permet l'impression de tous les caractères, y compris les caractères graphiques du CPC. Les images réalisées en graphisme haute résolution peuvent également être imprimées avec cette routine. Nous avons choisi comme imprimante la NLQ 401. Cette imprimante bon marché est, en ce qui concerne son jeu de caractères de commande, étonnamment compatible avec les imprimantes Epson MX/RX/FX. Les deux programmes tournent donc également sans adaptation sur des imprimantes Epson (et sur toutes les autres imprimantes compatibles).

A la fin de ce chapitre, vous ne trouverez pas uniquement deux routines de hardcopy rapides mais vous aurez également une première approche des routines du système d'exploitation.

Pour sortir le contenu de l'écran sur une imprimante connectée, il faut faire lire les caractères ligne par ligne sur l'écran et les sortir. Du fait de la structure spéciale de la Ram vidéo, il n'est malheureusement pas possible de lire les caractères directement.

A travers le 'détour' par une routine du système d'exploitation, il est cependant possible de déterminer quel caractère se trouve dans l'emplacement actuel du curseur. Cette routine (TXT RD CHAR, &BB60) transmet le caractère dans l'accumulateur et met le flag carry lorsqu'un caractère a été trouvé. Si par contre aucun caractère du jeu de caractères du CPC ne figure dans l'emplacement du curseur, alors l'accumulateur contient 0 et le flag carry est nul.

Il faut en outre une routine qui nous permette de positionner le curseur, de façon à ce que nous puissions lire les caractères les uns après les autres. Cette fonction est exécutée par TXT SET CURSOR, &BB75. Lorsque cette adresse est appelée, le contenu du registre H est interprété comme colonne et celui de L comme ligne. L'emplacement d'écriture supérieur gauche peut donc être ainsi adressé par &0101.

Il se pose ici cependant un petit problème. Après que nous ayons fait parcourir toute la surface de l'écran à notre curseur, avec l'interrogation de l'écran, il faudrait qu'il revienne ensuite dans son emplacement initial. Il nous faut donc pour cela, avant le premier positionnement du curseur, déterminer et ranger l'emplacement du curseur. Cela peut se faire grâce à TXT GET CURSOR, &BB78. Après avoir appelé TXT GET CURSOR le double registre HL contient la position actuelle du curseur. Il nous faut ranger cette valeur et la restaurer à la fin du hardcopy.

Les caractères obtenus grâce à TXT RD CHAR doivent être sortis sur l'imprimante. Nous pouvons utiliser à cet effet MC SEND PRINTER dont l'entrée est en &BD31. Le caractère figurant dans l'accumulateur est sorti avec sur le port d'imprimante avec tous les signaux handshake nécessaires.

MC SEND PRINTER attend toutefois que l'imprimante soit prête à recevoir. C'est MC BUSY PRINTER, &BD2E, qui nous permet de constater si c'est le cas. Si l'imprimante n'est pas prête à recevoir, si elle n'est pas allumée ou si elle n'est même pas connectée, MC BUSY PRINTER revient avec un flag carry mis. Dans ce cas, elle doit être appelée à nouveau, jusqu'à ce que le flag carry soit supprimé. Le caractère voulu peut alors être sorti.

Il peut cependant également arriver qu'un hardcopy une fois lancé ne doive pas être imprimé jusqu'au bout. L'opération peut être interrompue en appuyant sur la touche 'DEL'. Mais pour cela, il nous faut pouvoir examiner si cette touche est enfoncée. Si KM TEST KEY, &BB1E, est appelée avec un code de touche valable dans l'accumulateur, après exécution de cette routine, le flag zéro est nul si la touche correspondante est enfoncée. Sinon le flag zéro est mis.

Ainsi avons-nous en fait toutes les routines système nécessaires pour

écrire une routine de hardcopy. Mais nous nous rendrons compte au plus tard lorsque nous aurons commencé à écrire notre programme, que nous ne savons absolument pas si, au moment du hardcopy, il s'agit de représenter 20, 40 ou 80 caractères par ligne.

Bon, on pourrait décider que ce hardcopy ne fonctionne qu'en mode d'écran x. Mais ce serait une limitation peu élégante.

SCR GET MODE avec entrée en &BC11 nous communique avec l'accumulateur et les deux flags carry et zéro, dans que mode écran le CPC se trouve actuellement. Nous pouvons ainsi réaliser un hardcopy avec le nombre de caractères qui convient, en fonction des informations ainsi obtenues.

Mais venons-en maintenant au programme lui-même. Les lecteurs n'ayant pas d'assembleur peuvent utiliser le programme Basic imprimé à la fin de ce chapitre. Il contient les deux programmes de hardcopy en lignes de Data.

```
A100          ORG  #A100
BB78          GETCRS EQU #BB78
BB75          SETCRS EQU #BB75
BB60          RDCHAR EQU #BB60
BD2E          TSTPTR EQU #BD2E
BD31          PRTCHR EQU #BD31
BC11          GETMOD EQU #BC11
BB1E          TSTKEY EQU #BB1E
```

```
A100 CD78BB          CALL GETCRS      ;ranger ancienne position curseur
A103 2264A1          LD (OLDPOS),HL
A106 CD11BC          CALL GETMOD      ;chercher mode ecran
A109 17              RLA ;
                    ; nombre de caracteres/20
A10A 3263A1          LD (MODE),A ;et ranger
A10D 210101          LD HL,#0101 ;dans angle supérieur gauche
A110 2266A1          LD (CRSPOS),HL ;le curseur
A113 3A63A1 LL1     LD A,(MODE)
A116 47              LD B,A ;1,2 ou 4 fois
A117 0E14 LOOP      LD C,20 ;20 caractères par ligne
A119 C5 LLOOP       PUSH BC
A11A E5              PUSH HL
A11B CD75BB          CALL SETCRS      ;placer le curseur
A11E E1              POP HL
A11F CD60BB          CALL RDCHAR      ;et déterminer
```

```

A122 C1          POP BC      ;le caractère
A123 3802       JR C,GOOD   ;caractère valable?
A125 3E20       LD A,32     ;sinon sortir
A127 CD58A1 GOOD CALL PRTOUT ;espace
A12A E5        PUSH HL
A12B C5        PUSH BC
A12C 3E42       LD A,66     ;ESC enfoncée?
A12E CD1EBB     CALL TSTKEY
A131 C1        POP BC
A132 E1        POP HL
A133 201C       JR NZ,EXIT  ;si oui, fin
A135 24        WEITER INC H
A136 0D        DEC C
A137 20E0       JR NZ,LLOOP ;20 caractères imprimés?
A139 10DC       DJNZ LOOP  ;ligne entière?
A13B 3E0D       LD A,#0D    ;sortir CR/LF
A13D CD58A1     CALL PRTOUT
A140 3E0A       LD A,#0A
A142 CD58A1     CALL PRTOUT
A145 2A66A1     LD HL,(CRSPOS);déterminer
A148 2C        INC L      ;position curseur
A149 2266A1     LD (CRSPOS),HL;pour ligne suivante
A14C 7D        LD A,L
A14D FE1A       CP 26      ;25 lignes imprimées?
A14F 20C2       JR NZ,LL1
A151 2A64A1 EXIT LD HL,(OLDPOS);si oui, restaurer
A154 CD75BB     CALL SETCRS ;ancienne position curseur
A157 C9        RET ;      ;et retour
A158 C5        PRTOUT PUSH BC
A159 CD2EBD P1  CALL TSTPTR ;printer busy?
A15C 38FB       JR C,P1
A15E CD31BD     CALL PRTCHR ;sortir un caractère
A161 C1        POP BC
A162 C9        RET
A163 00        MODE  DEFB 0
A164 0000       OLDPOS DEFW 0000
A166 0000       CRSPOS DEFW 0000

```

Les commentaires dans le listing devraient rendre le programme facilement compréhensible. La seule particularité est constituée par la méthode de

calcul du nombre de caractères à sortir par ligne. C'est pourquoi nous voudrions évoquer cette question brièvement.

Après que nous ayons appelé SCR GET MODE, l'accumulateur contient, suivant le mode, 0, 1 ou 2. En outre les flags carry et zéro ont les états suivants:

Mode 0 = Carry 1, Zero 0

Mode 1 = Carry 0, Zero 1

Mode 2 = Carry 0, Zero 0

L'instruction SLA décale le contenu de l'accumulateur d'un bit vers la gauche. Cela correspond à une multiplication par deux. L'état du flag carry est en outre transféré dans le bit 0 de l'accumulateur et le bit 7 qui a été 'expulsé' est placé dans le carry.

En mode 0, le 0 qui se trouve dans l'accumulateur subit une rotation. Cela n'a pas d'influence sur le contenu de l'accumulateur. Mais comme le flag carry qui a été mis par SCR GET MODE est transféré dans le bit 0 de l'accumulateur, l'accumulateur contient 1 après cette instruction. Ce 1 a pour effet que une fois 20 caractères seront imprimés par ligne.

En mode 1, l'accumulateur contient un 1, le carry est nul dans ce mode. Après SLA, l'accumulateur contient un 2. Ce sont donc deux fois 20 caractères qui seront sortis par ligne. Le fonctionnement est analogue en mode 2. Le résultat de SLA est un 4 dans l'accumulateur, ce qui entraîne 4 fois 20 caractères par ligne d'impression.

Le principe est quelque peu différent quand il s'agit de produire un hardcopy graphique. Nous ne pouvons pas alors utiliser les routines TXT SET CURSOR et TXT RD CHAR.

Tout d'abord, GRA INITIALISE active le mode graphique. Ensuite, avec GRA GET PAPER nous déterminons le numéro de couleur du fond. Tous les points de l'écran seront comparés à cette valeur. Si la couleur d'un pixel est différente de celle du fond, un point sera produit sur le papier.

Malheureusement, le CPC ne dispose que d'une connexion 7 bits avec l'imprimante. Il en résulte certaines complications.

Cela signifie d'abord que nous pouvons sortir en une fois sur l'imprimante 7 points placés les uns sous les autres. Le graphisme du CPC a en tout une résolution graphique verticale de 200 points. Mais divisé par 7, cela ne donne pas une valeur entière. Il y a donc un reste, c'est-

à-dire des lignes de pixels qui devront être traitées d'une façon particulière. Le problème est cependant identique, quel que soit le mode de texte.

La sortie 7 bits pose un autre problème pour la transmission des instructions à l'imprimante. L'activation du graphisme avec ESC L nécessite pour les 640 pixels par ligne une indication qui ne peut être transmise par le CPC. Pour obtenir le nombre voulu de points graphiques sur l'imprimante, la séquence de commande pour l'imprimante est:

```
PRINT #8,CHR$(27);"L";CHR$(128)CHR$(2)
```

Le problème vient de la valeur 128. Exprimé en terme binaire, 128 est un nombre dont le huitième bit (le bit 7) est mis. Tous les autres bits sont nuls. Si nous envoyions cette valeur sur l'imprimante, celle-ci ne recevrait qu'un 0, puisque le huitième bit est utilisé comme strobe et n'est pas sorti vers l'imprimante.

Nous avons contourné ce problème de façon pas très élégante, en ne sortant horizontalement que 639 points. C'est certes un point de moins qu'il n'y en a sur l'écran, mais nous réduisons ainsi la première valeur à transmettre à 127 (maximum).

Avant que nous n'en venions maintenant au listing du hardcopy graphique, il nous faut encore relever une particularité.

Bien que l'écran ne représente physiquement que 200 lignes de grille, toutes les routines graphiques du CPC raisonnent à partir d'une résolution graphique de 400 points. Il en résulte un meilleur rapport entre les directions X et Y que si l'on ne comptait que les deux lignes véritablement existantes.

La conséquence est facile à observer si vous essayez par exemple le programme de dessin d'un cercle qui vous est proposé dans le manuel du CPC. Vous voyez en effet que le cercle est presque rond. Sans cette correction, c'est une ellipse allongée dans le sens de la largeur qui serait produite.

Cette correction doit également figurer dans notre hardcopy, mais sous une forme exactement contraire. Nous devons également déterminer les coordonnées graphique dans la grille de 400x640 points, mais sur l'imprimante, nous ne sortons que 200 points verticalement, pour ne pas avoir de gaspillages trop importants.

A000		ORG	#A000	
BBBA		GRINIT	EQU	#BBBA
BBE7		GETPAP	EQU	#BBE7
BBF0		TSTPOI	EQU	#BBF0
BD2B		PRINTO	EQU	#BD2B
BD2E		TSTPTR	EQU	#BD2E
BB1E		TSTKEY	EQU	#BB1E
A000	CDBABB	CALL	GRINIT	;activer mode graphique
A003	CDE7BB	CALL	GETPAPER	;déterminer couleur fond
A006	32BDA0	LD	(PAPER),A	
A009	CD6CA0	CALL	INITP	;fixer imprimante sur 7/72
A00C	218F01	LD	HL,399	;nous commençons
A00F	22BEA0	LD	(Y-MERK),HL	;l'impression
A012	110000	LD	DE,0	;en haut et à gauche
A015	3E07	LD	A,7	;mais avec malheureusement
A017	32C0A0	LD	(ANZAHL),A	;seulement 7 aiguilles
A01A	CD7CA0	LLOOP	CALL	PRTESC ;séquence ESC pour graphisme
A01D	0E00	LL1	LD	C,0 ;C contient modèle bits pour
A01F	3AC0A0		LD	A,(ANZAHL) ;l'imprimante
A022	47		LD	B,A ;B=compteur de lignes dot
A023	E5	BYTLP	PUSH	HL
A024	D5		PUSH	DE
A025	C5		PUSH	BC
A026	CDF0BB		CALL	TSTPOINT ;déterminer couleur du pixel
A029	C1		POP	BC ;d'emplacement (hl/de)
A02A	D1		POP	DE
A02B	21BDA0		LD	HL,PAPER
A02E	BE		CP	(HL) ;couleur pixel=couleur fond?
A02F	E1		POP	HL
A030	37		SCF	;
A031	2001		JR	NZ,DOT ;si pixel <> paper, alors
A033	A7		AND	A ;mettre flag carry, sinon
A034	CB11	DOT	RL	C ;annuler flag carry
A036	2B		DEC	HL ;décaler carry dans bit inférieur
A037	2B		DEC	HL ;du registre C
A038	10E9		DJNZ	BYTLP ;HL=HL-2, point suivant,
A03A	CDAFA0		CALL	TEST ;et le tout 7 fois
A03D	79		LD	A,C ;transférer dans accu
				;traitement spécial du dernier

A03E	CDA6A0	CALL	PRINT	;modèle bits et imprimer
A041	13	INC	DE	
A042	E5	PUSH	HL	
A043	217F02	LD	HL,639	;une ligne imprimée
A046	37	SCF		
A047	ED52	SBC	HL,DE	
A049	E1	POP	HL	
A04A	3805	JR	C,NXTROW	
A04C	2ABEA0	LD	HL,(Y-MERK)	
A04F	18CC	JR	LL1	
A051	23	NXTROW INC	HL	;traitement spécial des
A052	7C	LD	A,H	;4 dernières
A053	B5	OR	L	
A055	2B	DEC	HL	
A056	110000	LD	DE,0	;préparation de la prochaine
A059	22BEA0	LD	(Y-MERK),HL	;ligne d'impression
A05C	3E07	LD	A,7	
A05E	BD	CP	L	;dernière ligne de ??
A05F	20B9	JR	NZ,LLOOP	
A061	7C	LD	A,H	
A062	B4	OR	H	
A063	20B5	JR	NZ,LLOOP	
A065	3E04	LD	A,4	;alors plus que 4 lignes
A067	32C0A0	LD	(ANZAHL),A	
A06A	18AE	JR	LLOOP	
A06C	3E1B	INITP LD	A,27	;pour NLQ/MX/RX/FX
A06E	CDA6A0	CALL	PRINT	;ESC A 7, pour obtenir
A071	3E41	LD	A,65	;le bon passage à la ligne
A073	CDA6A0	CALL	PRINT	
A076	3E07	LD	A,7	
A078	CDA6A0	CALL	PRINT	
A07B	C9	RET		
A07C	E5	PRTESC PUSH	HL	;Touche DEL enfoncée?
A07D	3E42	LD	A,66	;si oui, alors interrompre HC
A07F	CD1EBB	CALL	TSTKEY	
A082	E1	POP	HL	
A083	2802	JR	Z,NOKEY	;DEL n'était pas enfoncée
A085	E1	POP	HL	;manipuler pile pour
A086	C9	RET	;	;atteindre le RET

```

A087 3E0D NOKEY LD A,#0D ;sortir CR/LF
A089 CDA6A0 CALL PRINT
A08C 3E0A LD A,10
A08E CDA6A0 CALL PRINT
A091 3E1B LD A,27 ;ESC L 127 2=graphisme
A093 CDA6A0 CALL PRINT ;avec 639 points
A096 3E4C LD A,76
A098 CDA6A0 CALL PRINT
A09B 3E7F LD A,127
A09D CDA6A0 CALL PRINT
A0A0 3E02 LD A,2
A0A2 CDA6A0 CALL PRINT
A0A5 C9 RET
A0A6 CD2EBD PRINT CALL TSTPTR ;imprimante busy?
A0A9 38FB JR C,PRINT
A0AB CD2BBD CALL PRINTOUT ;imprimer un caractère
A0AE C9 RET
A0AF 3AC0A0 TEST LD A,(ANZAHL) ;traitement des 4 dernières
A0B2 FE07 CP 7 ;lignes de dot
A0B4 C8 RET Z
A0B5 AF XOR A
A0B6 CB11 RL C ;décaler trois fois 0
A0B8 CB11 RL C ;dans le reg. C
A0BA CB11 RL C ;à travers carry
A0BC C9 RET
A0BD 00 PAPER DEFB 0
A0BE 0000 Y-MERK DEFW 0000
A0C0 00 ANZAHL DEFB 0

```

Voici enfin le programme de chargement en Basic que nous vous avons promis. Ce programme vous permet d'utiliser nos programmes, même si vous ne disposez pas d'un moniteur ou d'un assembleur.

Entrez d'abord la première partie du programme qui contient les messages et les commentaires:

```

100 REM Hardcopy graphique pour le CPC 464 avec NLQ/MX/RX/FX
110 REM Le hardcopy doit être appelé avec 'CALL &A000'
120 REM Hardcopy de texte pour le CPC 464

```



```
130 REM Le hardcopy doit etre appele avec 'CALL &A100'  
280 IF s<>23767 THEN PRINT"erreur dans hc graphique":END  
290 PRINT"Chargement de hc graphique correct"  
390 IF s<>11873 THEN PRINT"erreur dans hc de texte":END  
400 PRINT"Chargement de hc de texte correct"
```

```
140 FOR i = &a000 TO &a0bf  
150 READ byte : POKE i,byte : s = s + byte : NEXT  
160 DATA &CD,&BA,&BB,&CD,&E7,&BB,&32,&BD  
165 DATA &A0,&CD,&6C,&A0,&21,&8F,&01,&22  
170 DATA &BE,&A0,&11,&00,&00,&3E,&07,&32  
175 DATA &C0,&A0,&CD,&7C,&A0,&0E,&00,&3A  
180 DATA &C0,&A0,&47,&E5,&D5,&C5,&CD,&F0  
185 DATA &BB,&C1,&D1,&21,&BD,&A0,&BE,&E1  
190 DATA &37,&20,&01,&A7,&CB,&11,&2B,&2B  
195 DATA &10,&E9,&CD,&AF,&A0,&79,&CD,&A6  
200 DATA &A0,&13,&E5,&21,&7F,&02,&37,&ED  
205 DATA &52,&E1,&38,&05,&2A,&BE,&A0,&18
```

210 DATA &CC,&23,&7C,&B5,&C8,&2B,&11,&00
215 DATA &00,&22,&BE,&A0,&3E,&07,&BD,&20
220 DATA &B9,&7C,&B4,&20,&B5,&3E,&04,&32
225 DATA &C0,&A0,&18,&AE,&3E,&1B,&CD,&A6
230 DATA &A0,&3E,&41,&CD,&A6,&A0,&3E,&07
235 DATA &CD,&A6,&A0,&C9,&E5,&3E,&42,&CD
240 DATA &1E,&BB,&E1,&28,&02,&E1,&C9,&3E
245 DATA &0D,&CD,&A6,&A0,&3E,&0A,&CD,&A6
250 DATA &A0,&3E,&1B,&CD,&A6,&A0,&3E,&4C
255 DATA &CD,&A6,&A0,&3E,&7F,&CD,&A6,&A0
260 DATA &3E,&02,&CD,&A6,&A0,&C9,&CD,&2E
265 DATA &BD,&38,&FB,&CD,&2B,&BD,&C9,&3A
270 DATA &C0,&A0,&FE,&07,&C8,&AF,&CB,&11
275 DATA &CB,&11,&CB,&11,&C9,&00,&00,&00

300 FOR i = &a100 TO &a162 : s = 0
310 READ byte : POKE i,byte : s = s + byte : NEXT
320 DATA &CD,&78,&BB,&22,&64,&A1,&CD,&11
325 DATA &BC,&17,&32,&63,&A1,&21,&01,&01
330 DATA &22,&66,&A1,&3A,&63,&A1,&47,&0E
335 DATA &14,&C5,&E5,&CD,&75,&BB,&E1,&CD
340 DATA &60,&BB,&C1,&38,&02,&3E,&20,&CD
345 DATA &58,&A1,&E5,&C5,&3E,&42,&CD,&1E
350 DATA &BB,&C1,&E1,&20,&1C,&24,&0D,&20
355 DATA &E0,&10,&DC,&3E,&0D,&CD,&58,&A1
360 DATA &3E,&0A,&CD,&58,&A1,&2A,&66,&A1
365 DATA &2C,&22,&66,&A1,&7D,&FE,&1A,&20
370 DATA &C2,&2A,&64,&A1,&CD,&75,&BB,&C9
375 DATA &C5,&CD,&2E,&BD,&38,&FB,&CD,&31
380 DATA &BD,&C1,&C9

2.4 Le traitement des interruptions dans le système d'exploitation

La possibilité la plus rapide et la plus puissante de réagir à l'intérieur d'un système d'exploitation à certains événements est sans doute la technique des interruptions.

Vous savez certainement ce que c'est. Sinon, voici l'essentiel de ce qu'on peut dire à ce sujet:

Une interruption est en général un événement d'ordre électronique qui informe un programme en train de tourner qu'il vient de se produire. En fonction de cet événement, le logiciel doit entreprendre des actions correspondantes et ce le plus vite possible, suivant le niveau d'urgence. Une telle action sera par exemple le scrolling de l'écran pendant la phase sombre du rayon électronique, de façon à ce que l'image soit le plus nette possible.

Cette technique d'interruption présente l'avantage de n'interrompre le déroulement du reste du programme que lorsqu'il y a vraiment une action à effectuer, de sorte que le logiciel n'est pas constamment obligé de contrôler s'il se passe ou non quelque chose.

Il y a naturellement de nombreuses possibilités pour intégrer une telle fonction dans un système d'exploitation mais nous devons reconnaître que nous n'avions encore jamais rencontré une variante du type de celle qui fonctionne sur le CPC.

Il s'agit ici d'un mélange raffiné de hardware interrupt (interruption lorsque nécessaire) et de polling (examen régulier de ce qui se passe). Le programmeur de la routine correspondante décide du niveau d'urgence d'une 'demande'. En clair:

Il n'y a qu'une seule interruption dans la machine, le timer (appelé fast ticker dans le système), qui produit une interruption tous les 300èmes de seconde. Tout le reste en découle, comme vous allez voir.

Il est maintenant temps d'introduire quelques concepts que vous rencontrerez souvent à partir de maintenant, y compris dans le listing de la Rom.

1. EVENT signifie tout simplement événement. Comprenez qu'il s'agit d'une sorte d'interruption commandée par logiciel.
2. FRAME FLYBACK n'est rien d'autre que le retour déjà évoqué du rayon de l'écran, ce qui se produit tous les cinquantièmes de seconde.
3. TICKER est un multiple du fast ticker qui apparaît également tous les cinquantièmes de seconde.

Le tout est traité de façon à ce que le programmeur, donc éventuellement vous-même, quelles routines de son programme devront être appelées automatiquement, sans aucune intervention supplémentaire, et avec quelle fréquence elles devront être appelées au moment frame flyback, ticker ou même fast ticker. Comme préparation, il suffit, outre quelques petits détails, de communiquer une fois l'adresse de cette ou de ces routines. Cette information à préparer s'appelle EVENT BLOCK. Ici est indiqué avec quelle fréquence et quand la routine doit être appelée, si elle est ou non prioritaire par rapport à d'autres routines, etc...

À l'entrée du Ticker, Fast Ticker ou Frame Fly, le système d'exploitation regarde s'il y a des Event-Blocks correspondants. Si oui, ils sont appelés, en fonction de leur degré de priorité. Certains Event-Blocks existent en permanence, comme par exemple l'action qui consiste à alimenter le registre de couleur au moment Frame Fly.

Les blocs affectés à un évènement déterminé sont également reliés ensemble par le pointeur, de sorte que le système d'exploitation peut osciller de l'un à l'autre. Il est donc sans importance de savoir à quelle adresse figure un tel bloc, tant qu'il se trouve dans les 32K centraux de la Ram. Cette petite réserve doit être faite car cette zone est la seule à laquelle il soit possible d'accéder en permanence, indépendamment de la configuration de la Rom.

Si un tel bloc doit être exécuté, il est rangé dans ce qu'on appelle Pending Queue. Ce procédé est appelé Kicking.

La Pending Queue est traitée à la fin de la routine d'interruption propre du système. Vous vous dites certainement qu'un bloc existant doit naturellement être exécuté. Pourquoi donc faut-il le ranger dans une queue?

En fait les choses ne sont pas aussi simples car vous avez tout à fait la possibilité de suspendre le traitement d'un bloc pour un certain temps, sans que vous ayez à l'extraire de la queue primaire; ceci est d'ailleurs très facile à réaliser avec les Event-Blocks de la Ticker-Queue.

À propos: ne croyez pas qu'il n'y ait que cette interruption dans l'ordinateur. Les fanas de l'électronique ont tout à fait la possibilité de produire une interruption à travers le bus d'extension (asynchrone), mais il faut bien sûr qu'il y ait une routine correspondante qui puisse 'kicker' l'Event-Block' correspondant.

Devenons plus concret. Que faut-il faire lorsque vous voulez utiliser ce

mécanisme?

Il faut bien sûr commencer par créer un Event-Block dont la structure est définie ci-après. La partie suivante est commune à toutes les sortes d'évènements:

Octet 0+1 Adresse de chaîne pour la Pending Queue. Ce champ ne doit être alimenté que par le système d'exploitation!

Octet 2 Compteur

Tant que le compteur est > 0 , le bloc reste dans la Pending Queue, c'est-à-dire que la routine est exécutée jusqu'à ce qu'il soit égal à 0.

Si le compteur est < 0 (c'est-à-dire > 127), le bloc reste dans la chaîne correspondante (Ticker etc...). Le kicking ne conduit pas non plus dans ce cas à une exécution de la routine, alors que cela aurait normalement pour effet d'augmenter le compteur et donc de provoquer un saut à la prochaine occasion.

Octet 3 Classe

Bit0 = 1 = L'adresse de saut est une Near Address, c'est-à-dire qu'elle se trouve dans la Ram centrale ou dans la Rom inférieure.

Bit0 = 0 = L'adresse de saut est une Far Address, donc à rechercher dans la Rom supérieure.

Les bits 1-4 déterminent la priorité.

Bit 5 doit toujours être =0!

Bit6 = 1 = Express. Les Express-Events ont une priorité supérieure à celles des évènements normaux de la plus grande priorité.

Bit7 = 1 = Asynchron Event. Ces évènements n'ont pas de file d'attente et ils sont rangés immédiatement dans l'Interrupt Pending Queue lors du kicking (KL EVENT). S'il s'agit même d'un express, cette routine est exécutée immédiatement, sinon seulement à la fin de la routine d'interruption.

Attention: la routine pour les évènements asynchrones doit absolument se trouver dans la Ram centrale!

Octet4+5 Adresse de la routine

Octet 6 Rom Select, si l'adresse de saut est du type Far, sinon inutilisé.

Octet 7 Ici commence le champ de l'utilisateur qui peut être aussi long que souhaité. Il peut servir à la transmission de paramètres à la routine. Lors de l'appel d'une Event-Routine hl contient l'adresse de l'octet 5 de l'Event Block, s'il s'agit d'une Near

Address, sinon l'adresse de l'octet 6.

Ceci permet de créer plusieurs blocs pour une même routine qui peut déterminer, en fonction des paramètres, par quel bloc elle a été appelée.

Suivant le type de l'évènement, Ticker, Fast Ticker ou Frame Fly, deux ou six octets sont encore placés avant la partie commune. Dans le cas de Fast Ticker et Frame Fly, ce ne sont que deux octets pour le chaînage (ne pas les modifier!) dans la Fast Ticker List ou la Frame Fly List.

Les six octets pour le Ticker ont la signification suivante:

Octet 0+1 Chaînage pour Ticker List (ne pas modifier!)

Octet 2+3 Tick Count détermine combien de fois un Ticker doit apparaître, avant que le bloc ne soit kické une fois.

Octet 4+5 Reload Count indique quelle valeur doit être chargée dans le Tick Count après son écoulement.

Après donc que vous ayez alimenté votre bloc avec ces valeurs, pour autant que vous les connaissiez, (ce devraient être les 5 derniers octets (Event Count=0) de la partie commune et, pour le ticker, également les compteurs), vous n'avez plus qu'à charger l'adresse de début de votre bloc dans hl puis, suivant le cas, à appeler la routine KL ADD TICKER, KL ADD FAST TICKER ou KL ADD FRAME FLY.

Pour extraire le bloc de la liste, utilisez les routines KL DEL TICKER, etc... hl devant cette fois également contenir l'adresse du bloc à éloigner.

Essayez et observez comment le système d'exploitation procède, car les procédures qui reviennent sans cesse sont également traitées à travers le mécanisme des évènements.

2.5 Le listing de la Rom du système d'exploitation

Nous nous sommes donnés le plus grand mal pour que vous puissiez utiliser le plus aisément possible ce listing de la Rom, mais il reste encore des blancs sur notre carte d'état-major, essentiellement d'ailleurs là, où il ne s'agit pas de la structure du système en tant que telle, mais où certaines fonctions particulières sont exécutées. Il s'agit par exemple du CASSETTE MANAGER, du GRAPHICS MANAGER et du SOUND MANAGER. De tels programmes sont naturellement difficiles à interpréter car il est impossible de reconstituer le processus de pensée de chaque programmeur. Mais nous pensons que cela ne devrait pas vous gêner dans l'utilisation des routines.

Vous trouverez dans l'introduction à chaque pack des indications pour appeler certaines sections de programme souvent utilisées avec les paramètres à transmettre.

Les paramètres à transmettre de toutes routines dotées de vecteurs, qu'elles soient utilisables ou non figurent dans le Schneider Firmware Manual. C'est de la version anglaise de ce manuel que nous avons tiré les noms des packs que nous n'avons pas traduit pour éviter toute confusion dans l'esprit des lecteurs possédant ce manuel.

2.5.1 KERNEL (KL)

Le Kernel, comme son nom l'indique est le noyau du système d'exploitation.

C'est ainsi qu'il est responsable de la commande du déroulement des programmes, c'est-à-dire pour le traitement des interruptions ainsi que des Events, le traitement des Restarts, la mise en place d'extensions de la Rom et la commutation de la configuration de la mémoire.

Les routines liées au mécanisme des Events sont éventuellement utilisables. Voyez à ce sujet le chapitre 2.4.

KERNEL

```

0000 01897F      ld  bc,7fB9      U Rom dis., Mode 1, res diviseur
0003 ED49        out  (c),c
0005 C38005      jp   0580        RESET CONT'D

0008 C382B9      jp   B982 (0413) RST 1 LOW JUMP CONT'D

000B C37CB9      jp   B97C (040D) KL LOW PCHL CONT'D

000E C5          push bc
000F C9          ret              jp (bc)

0010 C316BA      jp   BA16 (04A7) RST 2 LOW SIDE CALL CONT'D

0013 C310BA      jp   BA10 (04A1) KL SIDE PCHL CONT'D

0016 D5          push de
0017 C9          ret              jp (de)

0018 C3BFB9      jp   B9BF (0450) RST 3 LOW FAR CALL CONT'D

001B C3B1B9      jp   B9B1 (0442) KL FAR PCHL CONT'D

001E E9          jp   (h1)

001F 00          nop
0020 C3CBBA      jp   BACB (055C) RST 4 RAM LAM CONT'D

0023 C3B9B9      jp   B9B9 (044A) KL FAR ICALL CONT'D

0026 00          nop
0027 00          nop
0028 C32EBA      jp   BA2E (04BF) RST 5 FIRM JUMP CONT'D

002B 00          nop
002C ED49      out  (c),c
002E D9          exx
002F FB          ei

***** RST 6 USER0
0030 F3          di          RSTO après High Kernel Restore

```

KERNEL

```

0031 D9          exx
0032 212B00     ld   hl,002B
0035 71         ld   (hl),c
0036 1808       jr   0040
0038 C339B9     jp   B939 (03CA) RST 7 INTERRUPT ENTRY CONT'D

```

```

003B C9         ret           EXT INTERRUPT

```

```

003C 00         nop
003D 00         nop
003E 00         nop
003F 00         nop

```

***** Jusqu'ici copié dans la Ram

```

0040 CBD1       set   2,c           L Rom disable
0042 18E8       jr   002C

```

***** Restore High Kernel Jumps

```

0044 214000     ld   hl,0040     003f
0047 2D         dec   l           à
0048 7E         ld   a,(hl)     0000
0049 77         ld   (hl),a     copier dans
004A 20FB       jr   nz,0047    la Ram
004C 3EC7       ld   a,C7       RST 0 dans
004E 323000     ld   (0030),a   0030
0051 219103     ld   hl,0391    Jump
0054 1100B9     ld   de,B900 (0391) Copier
0057 01E901     ld   bc,01E9    bloc
005A EDB0       ldir

```

***** KL CHOKE OFF

```

005C F3         di
005D 3AABB1     ld   a,(B1AB)   (config. Rom act.)
0060 ED5BA9B1  ld   de,(B1A9) (Entrée Rom act.)
0064 06C0       ld   b,C0       Firmware-
0066 2100B1     ld   hl,B100    Ram
0069 3600       ld   (hl),00
006B 23         inc   hl         Supprimer
006C 10FB       djnz 0069       jusqu'à B1C0
006E 47         ld   b,a
006F 0EFF       ld   c,FF

```

KERNEL

```

0071 A9      xor  c      Y avait-il une Rom active?
0072 C0      ret  nz      oui >
0073 4F      ld  c,a
0074 5F      ld  e,a
0075 57      ld  d,a
0076 C9      ret

```

```

0077 7C      ld  a,h
0078 B5      or   l
0079 79      ld  a,c
007A 2004    jr  nz,0080
007C 7D      ld  a,l      si hl=0
007D 2106C0   ld  hl,C006   Chargement par défaut
0080 32A8B1   ld  (B1A8),a (Rom ext. act.)
0083 32ABB1   ld  (B1AB),a (config. Rom act.)
0086 22A9B1   ld  (B1A9),hl (Entrée Rom act.)
0089 21FFAB   ld  hl,ABFF   Charger params pour
008C 114000   ld  de,0040   RST3
008F 01FFB0   ld  bc,B0FF
0092 3100C0   ld  sp,C000
0095 DF      rst  3      FAR CALL
0096 A9B1    dw  B1A9
0098 C7      rst  0

```

***** KL TIME PLEASE

```

0099 F3      di
009A ED5B89B1 ld  de,(B189) (Timer high)
009E 2A87B1   ld  hl,(B187) (Timer low)
00A1 FB      ei
00A2 C9      ret

```

***** KL TIME SET

```

00A3 F3      di
00A4 AF      xor  a
00A5 328BB1   ld  (B18B),a (Timerflag)
00A8 ED5389B1 ld  (B189),de (Timer high)
00AC 2287B1   ld  (B187),hl (Timer low)
00AF FB      ei
00B0 C9      ret

```

***** Scan Events

KERNEL

00B1	2187B1	ld	hl,B187	Timer low
00B4	34	inc	(hl)	update
00B5	23	inc	hl	Timer
00B6	28FC	jr	z,00B4	
00B8	06F5	ld	b,F5	
00BA	ED78	in	a,(c)	Port B
00BC	1F	rra		VSYNC ?
00BD	3008	jr	nc,00C7	Non >
00BF	2A8CB1	ld	hl,(B18C)	(Start Frame Fly Chain)
00C2	7C	ld	a,h	
00C3	B7	or	a	
00C4	C45301	call	nz,0153	Kick Event
00C7	2A8EB1	ld	hl,(B18E)	(Start Fast Ticker Chain)
00CA	7C	ld	a,h	
00CB	B7	or	a	
00CC	C45301	call	nz,0153	Kick Event
00CF	CD611F	call	1F61	Scan Sound Queues
00D2	2192B1	ld	hl,B192	Count for Ticker'
00D5	35	dec	(hl)	
00D6	C0	ret	nz	
00D7	3606	ld	(hl),06	
00D9	CDB71B	call	1BB7	Update Key State Map
00DC	2A90B1	ld	hl,(B190)	(Start Ticker Chain)
00DF	7C	ld	a,h	
00E0	B7	or	a	
00E1	C8	ret	z	
00E2	2104B1	ld	hl,B104	div. flags pour rout. int.
00E5	CBC6	set	0,(hl)	Ticker Chain doit encore
00E7	C9	ret		être traité
00E8	2B	dec	hl	
00E9	3600	ld	(hl),00	
00EB	2B	dec	hl	
00EC	3A01B1	ld	a,(B101)	
00EF	B7	or	a	
00F0	200C	jr	nz,00FE	
00F2	2200B1	ld	(B100),hl	(Start Int Pending Queue)
00F5	2202B1	ld	(B102),hl	
00F8	2104B1	ld	hl,B104	div. flags pour rout. int.
00FB	CBF6	set	6,(hl)	

KERNEL

```

00FD C9          ret

00FE ED5B02B1   ld  de,(B102)
0102 2202B1     ld  (B102),hl
0105 EB         ex  de,hl
0106 73         ld  (hl),e
0107 23         inc hl
0108 72         ld  (hl),d
0109 C9         ret

010A ED7305B1   ld  (B105),sp  (sp save)
010E 3187B1     ld  sp,B187   Timer low
0111 E5         push hl
0112 D5         push de
0113 C5         push bc
0114 2104B1     ld  hl,B104   div. flags pour rout. int.
0117 CB76       bit  6,(hl)
0119 281E       jr  z,0139
011B CBFE       set  7,(hl)
011D 2A00B1     ld  hl,(B100) (Start Int Pending Queue)
0120 7C         ld  a,h
0121 B7         or  a
0122 280E       jr  z,0132
0124 5E         ld  e,(hl)
0125 23         inc hl
0126 56         ld  d,(hl)
0127 ED5300B1   ld  (B100),de (Start Int Pending Queue)
012B 23         inc hl
012C C0A02      call 020A
012F F3         di
0130 18EB       jr  011D
0132 2104B1     ld  hl,B104   div. flags pour rout. int.
0135 CB46       bit  0,(hl)   Ticker Queue pending ?
0137 2810       jr  z,0149   non
0139 3600       ld  (hl),00
013B 37         scf
013C 08         ex  af,af
013D CD8901     call 0189     traiter Ticker Chain
0140 B7         or  a
0141 08         ex  af,af'

```

KERNEL

```

0142 2104B1      ld  h1,B104      div. flags pour rout. int.
0145 7E         ld  a,(h1)
0146 B7         or  a           encore quelque chose à traiter?
0147 20D2       jr  nz,011B      oui
0149 3600       ld  (h1),00     supprimer tous les flags
014B C1         pop  bc
014C D1         pop  de
014D E1         pop  h1
014E ED7B05B1   ld  sp,(B105)   recharger sp
0152 C9         ret

```

***** Kick Event

```

0153 5E         ld  e,(h1)
0154 23         inc  h1
0155 7E         ld  a,(h1)
0156 23         inc  h1
0157 B7         or  a
0158 CAE201     jp  z,01E2      KL EVENT
015B 57         ld  d,a
015C D5         push de
015D CDE201     call 01E2      KL EVENT
0160 E1         pop  h1
0161 18F0       jr  0153      Kick Event

```

***** KL NEW FRAME FLY

```

0163 E5         push h1
0164 23         inc  h1
0165 23         inc  h1
0166 CDD201     call 01D2      KL INIT EVENT
0169 E1         pop  h1

```

***** KL ADD FRAME FLY

```

016A 118CB1     ld  de,B18C    Start Frame Fly Chain
016D C37303     jp  0373      Add Event

0170 118CB1     ld  de,B18C    Start Frame Fly Chain
0173 C38203     jp  0382      Delete Event

```

***** KL NEW FAST TICKER

```

0176 E5         push h1

```

KERNEL

```

0177 23          inc  hl
0178 23          inc  hl
0179 CDD201     call 01D2      KL INIT EVENT
017C E1         pop   hl

```

***** KL ADD FAST TICKERO

```

017D 118EB1     ld    de,B18E      Start Fast Ticker Chain
0180 C37303     jp    0373        Add Event

```

***** Delete Fast Ticker

```

0183 118EB1     ld    de,B18E      Start Fast Ticker Chain
0186 C38203     jp    0382        Delete Event

```

***** Ticker Chain bearbeiten

```

0189 2A90B1     ld    hl,(B190)    (Start Ticker Chain)
018C 7C         ld    a,h
018D B7         or    a
018E C8         ret   z
018F 5E         ld    e,(hl)
0190 23         inc  hl
0191 56         ld    d,(hl)
0192 23         inc  hl
0193 4E         ld    c,(hl)
0194 23         inc  hl
0195 46         ld    b,(hl)
0196 78         ld    a,b
0197 B1         or    c
0198 2816       jr    z,01B0
019A 0B         dec  bc
019B 78         ld    a,b
019C B1         or    c
019D 200E       jr    nz,01AD
019F D5         push de
01A0 23         inc  hl
01A1 23         inc  hl
01A2 E5         push hl
01A3 23         inc  hl
01A4 CDE201     call 01E2      KL EVENT
01A7 E1         pop   hl
01A8 46         ld    b,(hl)

```

KERNEL

```

01A9 2B      dec  hl
01AA 4E      ld   c,(hl)
01AB 2B      dec  hl
01AC D1      pop  de
01AD 70      ld   (hl),b
01AE 2B      dec  hl
01AF 71      ld   (hl),c
01B0 EB      ex   de,hl
01B1 18D9    jr   018C

```

***** KL ADD TICKER

```

01B3 E5      push hl
01B4 23      inc  hl
01B5 23      inc  hl
01B6 F3      di
01B7 73      ld   (hl),e
01B8 23      inc  hl
01B9 72      ld   (hl),d
01BA 23      inc  hl
01BB 71      ld   (hl),c
01BC 23      inc  hl
01BD 70      ld   (hl),b
01BE E1      pop  hl
01BF 1190B1  ld   de,B190   Start Ticker Chain
01C2 C37303  jp   0373      Add Event

```

***** Delete Ticker

```

01C5 1190B1  ld   de,B190   Start Ticker Chain
01C8 CD8203  call 0382      Delete Event
01CB D0      ret  nc
01CC EB      ex   de,hl
01CD 23      inc  hl
01CE 5E      ld   e,(hl)
01CF 23      inc  hl
01D0 56      ld   d,(hl)
01D1 C9      ret

```

***** KL INIT EVENT

```

01D2 F3      di
01D3 23      inc  hl

```


KERNEL

```

01D4 23      inc  hl
01D5 3600    ld   (hl),00
01D7 23      inc  hl
01D8 70      ld   (hl),b
01D9 23      inc  hl
01DA 73      ld   (hl),e
01DB 23      inc  hl
01DC 72      ld   (hl),d
01DD 23      inc  hl
01DE 71      ld   (hl),c
01DF 23      inc  hl
01E0 FB      ei
01E1 C9      ret

```

***** KL EVENT

```

01E2 23      inc  hl
01E3 23      inc  hl
01E4 F3      di
01E5 7E      ld   a,(hl)
01E6 34      inc  (hl)
01E7 FA0602  jp   m,0206      Event Cnt >127/<0
01EA B7      or   a
01EB 2013    jr   nz,0200     Event Cnt >0 & <127
01ED 23      inc  hl
01EE 7E      ld   a,(hl)
01EF 2B      dec  hl
01F0 B7      or   a
01F1 F22F02  jp   p,022F      ajouter Sync Event
01F4 08      ex  af,af'
01F5 3012    jr   nc,0209
01F7 08      ex  af,af'
01F8 87      add  a,a
01F9 F2E800  jp   p,00E8
01FC 23      inc  hl
01FD 23      inc  hl
01FE 1823    jr   0223
0200 08      ex  af,af'
0201 3801    jr   c,0204
0203 FB      ei
0204 08      ex  af,af'

```

KERNEL

```

0205 C9          ret
0206 35          dec  (hl)
0207 18F7        jr    0200
0209 08          ex   af,af'
020A FB          ei
020B 7E          ld   a,(hl)
020C B7          or   a
020D F8          ret  m
020E E5          push hl
020F CD1C02     call 021C
0212 E1          pop  hl
0213 35          dec  (hl)
0214 C8          ret  z
0215 F20E02     jp   p,020E
0218 34          inc  (hl)
0219 C9          ret

```

***** KL DO SYNC

```

021A 23          inc  hl
021B 23          inc  hl
021C 23          inc  hl
021D 7E          ld   a,(hl)
021E 23          inc  hl
021F 1F          rra
0220 D2B9B9     jp   nc,B9B9 (044A) KL FAR ICALL CONT'D
0223 5E          ld   e,(hl)
0224 23          inc  hl
0225 56          ld   d,(hl)
0226 EB          ex   de,hl
0227 E9          jp   (hl)

```

***** KL SYNC RESET

```

0228 210000     ld   hl,0000
022B 2294B1     ld   (B194),hl
022E C9          ret

```

***** Ajouter Sync Event

```

022F E5          push hl
0230 47          ld   b,a          Priorité => b

```

KERNEL

```

0231 1196B1      ld    de,B196
0234 EB         ex    de,hl
0235 2B         dec   hl
0236 2B         dec   hl
0237 56         ld    d,(hl)      Adr. prochain
0238 2B         dec   hl          Event Block
0239 5E         ld    e,(hl)      de
023A 7A         ld    a,d
023B B7         or    a
023C 2807       jr    z,0245
023E 13         inc   de
023F 13         inc   de
0240 13         inc   -de
0241 1A         ld    a,(de)      Prioritat act. >
0242 B8         cp    b          priorité trouvée?
0243 30EF       jr    nc,0234     non
0245 D1         pop   de
0246 1B         dec   de
0247 23         inc   hl
0248 7E         ld    a,(hl)
0249 12         ld    (de),a
024A 1B         dec   de
024B 72         ld    (hl),d
024C 2B         dec   hl
024D 7E         ld    a,(hl)
024E 12         ld    (de),a
024F 73         ld    (hl),e
0250 08         ex    af,af'
0251 3801       jr    c,0254
0253 FB         ei
0254 08         ex    af,af'
0255 C9         ret

```

***** KL NEXT SYNC

```

0256 F3         di
0257 2A93B1     ld    hl,(B193)   (Start Sync Pending Queue)
025A 7C         ld    a,h
025B B7         or    a
025C 2817       jr    z,0275
025E E5         push hl

```

KERNEL

```

025F 5E      ld  e,(hl)
0260 23      inc hl
0261 56      ld  d,(hl)
0262 23      inc hl
0263 23      inc hl
0264 3A95B1  ld  a,(B195)  (Priorité Event act.)
0267 BE      cp  (hl)
0268 300A    jr  nc,0274
026A F5      push af
026B 7E      ld  a,(hl)
026C 3295B1  ld  (B195),a  (Priorité Event act.)
026F ED5393B1 ld  (B193),de (Start Sync Pending Queue)
0273 F1      pop  af
0274 E1      pop  hl
0275 FB      ei
0276 C9      ret

```

***** KL DONE SYNC

```

0277 3295B1  ld  (B195),a  (Priorité Event act.)
027A 23      inc  hl
027B 23      inc  hl
027C 35      dec  (hl)
027D C8      ret  z
027E F3      di
027F F22F02  jp  p,022F    AJouter Sync Event
0282 34      inc  (hl)
0283 FB      ei
0284 C9      ret

```

***** KL DEL SYNCHRONOUS

```

0285 CD8E02  call 028E    KL DISARM EVENT
0288 1193B1  ld  de,B193  Start Sync Pending Queue
028B C38203  jp  0382    Delete Event

```

***** KL DISARM EVENT

```

028E 23      inc  hl
028F 23      inc  hl
0290 36C0    ld  (hl),C0
0292 2B      dec  hl
0293 2B      dec  hl

```

KERNEL

0294 C9 ret

***** KL EVENT DISABLE

0295 2195B1 ld hl,B195 Priorité Event act.
 0298 CBEE set 5,(hl)
 029A C9 ret

***** KL EVENT ENABLE

029B 2195B1 ld hl,B195 Priorité Event act.
 029E CBAE res 5,(hl)
 02A0 C9 ret

***** KL LOG EXT

02A1 E5 push hl
 02A2 ED5BA6B1 ld de,(B1A6)
 02A6 22A6B1 ld (B1A6),hl
 02A9 73 ld (hl),e
 02AA 23 inc hl
 02AB 72 ld (hl),d
 02AC 23 inc hl
 02AD 71 ld (hl),c
 02AE 23 inc hl
 02AF 70 ld (hl),b
 02B0 E1 pop hl
 02B1 C9 ret

***** KL FIND COMMAND

02B2 1196B1 ld de,B196 instruction à exécuter
 02B5 011000 ld bc,0010
 02B8 CDA6BA call BAA6 (0537) Rom off & save config
 02BB EB ex de,hl
 02BC 2B dec hl
 02BD CBFE set 7,(hl)
 02BF 2AA6B1 ld hl,(B1A6)
 02C2 7D ld a,l
 02C3 1810 jr 02D5
 02C5 E5 push hl
 02C6 23 inc hl
 02C7 23 inc hl
 02C8 4E ld c,(hl)

KERNEL

```

02C9 23      inc  hl
02CA 46      ld   b,(hl)
02CB CDF402  call 02F4
02CE D1      pop  de
02CF D8      ret  c
02D0 EB      ex  de,hl
02D1 7E      ld  a,(hl)
02D2 23      inc  hl
02D3 66      ld  h,(hl)
02D4 6F      ld  l,a
02D5 B4      or   h
02D6 20ED    jr   nz,02C5
02D8 0EFF    ld  c,FF
02DA 0C      inc  c
02DB CD83BA  call BA83 (0514) KL PROBE ROM CONT'D
02DE F5      push af
02DF E603    and  03
02E1 47      ld  b,a
02E2 CCF402  call z,02F4
02E5 3809    jr   c,02F0
02E7 F1      pop  af
02E8 87      add  a,a
02E9 30EF    jr   nc,02DA
02EB 79      ld  a,c
02EC B7      or   a
02ED 28EB    jr   z,02DA
02EF C9      ret

02F0 F1      pop  af
02F1 C30B06  jp   060B      MC START PROGRAM

02F4 2104C0  ld  hl,C004
02F7 78      ld  a,b
02F8 B7      or   a
02F9 2804    jr   z,02FF
02FB 60      ld  h,b
02FC 69      ld  l,c
02FD 0EFF    ld  c,FF
02FF CD7EBA  call BA7E (050F) KL ROM SELECT CONT'D
0302 C5      push bc

```

KERNEL

```

0303 5E      ld    e,(hl)
0304 23      inc   hl
0305 56      ld    d,(hl)
0306 23      inc   hl
0307 EB      ex    de,hl
0308 1817     jr    0321
030A 0196B1   ld    bc,B196      instruction à exécuter
030D 0A      ld    a,(bc)
030E BE      cp    (hl)
030F 2008     jr    nz,0319
0311 23      inc   hl
0312 03      inc   bc
0313 87      add   a,a
0314 30F7     jr    nc,030D
0316 EB      ex    de,hl
0317 180C     jr    0325
0319 7E      ld    a,(hl)
031A 23      inc   hl
031B 87      add   a,a
031C 30FB     jr    nc,0319
031E 13      inc   de
031F 13      inc   de
0320 13      inc   de
0321 7E      ld    a,(hl)
0322 B7      or    a
0323 20E5     jr    nz,030A
0325 C1      pop   bc
0326 C38CBA   jp    BABC (051D) KL ROM DESELECT CONT'D

```

***** KL ROM WALK

```

0329 0E07     ld    c,07
032B CD3203   call 0332      KL INIT BACK
032E 0D      dec   c
032F 20FA     jr    nz,032B
0331 C9      ret

```

***** KL INIT BACK

```

0332 79      ld    a,c
0333 FE08     cp    08
0335 D0      ret   nc

```

KERNEL

```

0336 CD7EBA      call BA7E (050F) KL ROM SELECT CONT'D
0339 3A00C0      ld    a,(C000)
033C E603        and   03
033E 3D          dec   a
033F 201F        Jr    nz,0360
0341 C5          push  bc
0342 CD06C0      call C006
0345 D5          push  de
0346 23          inc   hl
0347 EB          ex    de,hl
0348 21AAB1      ld    hl,B1AA
034B ED4BA8B1   ld    bc,(B1A8) (Rom. ext. act.)
034F 0600        ld    b,00
0351 09          add   hl,bc
0352 09          add   hl,bc
0353 73          ld    (hl),e
0354 23          inc   hl
0355 72          ld    (hl),d
0356 21FCFF     ld    hl,FFFC
0359 19          add   hl,de
035A CDA102     call 02A1      KL LOG EXT
035D 2B          dec   hl
035E D1          pop   de
035F C1          pop   bc
0360 C38CBA     Jp    BA8C (051D) KL ROM Deselect CONT'D

0363 7E          ld    a,(hl)
0364 BB          cp    e
0365 23          inc   hl
0366 7E          ld    a,(hl)
0367 2B          dec   hl
0368 2003        Jr    nz,036D
036A BA          cp    d
036B 37          scf
036C C8          ret   z
036D B7          or    a
036E C8          ret   z
036F 6E          ld    l,(hl)
0370 67          ld    h,a
0371 18F0        Jr    0363

```


KERNEL

***** Add Event

```

0373 EB      ex    de,hl
0374 F3      di
0375 CD6303  call  0363
0378 3806    jr    c,0380
037A 73      ld    (hl),e
037B 23      inc  hl
037C 72      ld    (hl),d
037D 13      inc  de
037E AF      xor  a
037F 12      ld    (de),a
0380 FB      ei
0381 C9      ret

```

***** Delete Event

```

0382 EB      ex    de,hl
0383 F3      di
0384 CD6303  call  0363
0387 3006    jr    nc,038F
0389 1A      ld    a,(de)
038A 77      ld    (hl),a
038B 13      inc  de
038C 23      inc  hl
038D 1A      ld    a,(de)
038E 77      ld    (hl),a
038F FB      ei
0390 C9      ret

```

```

0391 C35EBA    jp    BA5E (04EF) KL U ROM ENABLE CONT'D
0394 C368BA    jp    BA68 (04F9) KL U ROM DISABLE CONT'D
0397 C34ABA    jp    BA4A (04DB) KL L ROM ENABLE CONT'D
039A C354BA    jp    BA54 (04E5) KL L ROM DISABLE CONT'D
039D C372BA    jp    BA72 (0503) KL ROM RESTORE CONT'D
03A0 C37EBA    jp    BA7E (050F) KL ROM SELECT CONT'D

```

KERNEL

```

03A3 C3A2BA      Jp   BAA2 (0533) KL CURR SELECTION CONT'D
03A6 C383BA      Jp   BA83 (0514) KL PROBE ROM CONT'D
03A9 C38CBA      Jp   BA8C (051D) KL ROM DESELECT CONT'D
03AC C3A6BA      Jp   BAA6 (0537) KL LDIR CONT'D
03AF C3ACBA      Jp   BAAC (053D) KL LDDR CONT'D

```

***** KL POLL SYNCHRONOUS

```

03B2 3A94B1      ld   a,(B194)
03B5 B7           or   a
03B6 C8           ret  z
03B7 E5           push hl
03B8 F3           di
03B9 2A93B1      ld   hl,(B193) (Start Sync Pending Queue)
03BC 7C           ld   a,h
03BD B7           or   a
03BE 2807        jr   z,03C7
03C0 23           inc  hl
03C1 23           inc  hl
03C2 23           inc  hl
03C3 3A95B1      ld   a,(B195) (Priorité Event act.)
03C6 BE           cp   (hl)
03C7 E1           pop  hl
03C8 FB           ei
03C9 C9           ret

```

***** RST 7 INTERRUPT ENTRY CONT'D

```

03CA F3           di
03CB 08           ex   af,af'
03CC 3833        jr   c,0401      EXT INTERRUPT ENTRY
03CE D9           exx
03CF 79           ld   a,c
03D0 37           scf
03D1 FB           ei
03D2 08           ex   af,af'
03D3 F3           di

```

KERNEL

```

03D4 F5      push  af
03D5 CB91    res   2,c
03D7 ED49    out   (c),c      L Rom enable
03D9 CDB100  call  00B1      Scan Events
03DC B7       or    a
03DD 08      ex   af,af'
03DE 4F      ld   c,a
03DF 067F    ld   b,7F
03E1 3A04B1  ld   a,(B104)   (div. flags pour rout. int.)
03E4 B7       or    a
03E5 2814    jr   z,03FB
03E7 FA6AB9  jp   m,B96A (03FB)
03EA 79      ld   a,c
03EB E60C    and  UC
03ED F5      push  af
03EE CB91    res   2,c
03F0 D9      exx
03F1 CD0A01  call  010A
03F4 D9      exx
03F5 E1      pop   hl
03F6 79      ld   a,c
03F7 E6F3    and  F3
03F9 B4      or    h
03FA 4F      ld   c,a
03FB ED49    out   (c),c      fixer ancienne config.
03FD D9      exx
03FE F1      pop   af
03FF FB      ei
0400 C9      ret

```

***** EXT INTERRUPT ENTRY

```

0401 08      ex   af,af'
0402 E1      pop   hl
0403 F5      push  af
0404 CBD1    set   2,c
0406 ED49    out   (c),c      L Rom disable
0408 CD3B00  call  003B
040B 18CF    jr   03DC

```

***** KL LOW PCHL CONT'D

KERNEL

```

040D F3      di
040E E5      push hl
040F D9      exx
0410 D1      pop  de
0411 1806    jr    0419

```

***** RST 1 LOW JUMP CONT'D

```

0413 F3      di
0414 D9      exx
0415 E1      pop  hl
0416 5E      ld  e,(hl)
0417 23      inc hl
0418 56      ld  d,(hl)
0419 08      ex  af,af'
041A 7A      ld  a,d
041B CBBA    res  7,d
041D CBB2    res  6,d
041F 07      rlca
0420 07      rlca
0421 07      rlca
0422 07      rlca
0423 A9      xor  c
0424 E60C    and  0C
0426 A9      xor  c
0427 C5      push bc
0428 CDA8B9  call B9A8 (0439) préparer config. & exécuter saut
042B F3      di
042C D9      exx
042D 08      ex  af,af'
042E 79      ld  a,c
042F C1      pop  bc
0430 E603    and  03
0432 CB89    res  1,c
0434 CB81    res  0,c
0436 B1      or   c
0437 1801    jr   043A
0439 D5      push de      adr. de saut sur pile
043A 4F      ld  c,a
043B ED49    out (c),c    fixer config Rom
043D B7      or   a

```

KERNEL

```

043E 08      ex  af,af'
043F D9      exx
0440 FB      ei
0441 C9      ret          exécuter saut

```

***** KL FAR PCHL CONT'D

```

0442 F3      di
0443 08      ex  af,af'
0444 79      ld  a,c
0445 E5      push hl
0446 D9      exx
0447 D1      pop  de
0448 1815    jr   045F

```

***** KL FAR ICALL CONT'D

```

044A F3      di
044B E5      push hl
044C D9      exx
044D E1      pop  hl
044E 1809    jr   0459

```

***** RST 3 LOW FAR CALL CONT'D

```

0450 F3      di
0451 D9      exx
0452 E1      pop  hl
0453 5E      ld  e,(hl)
0454 23      inc  hl
0455 56      ld  d,(hl)
0456 23      inc  hl
0457 E5      push hl
0458 EB      ex  de,hl
0459 5E      ld  e,(hl)
045A 23      inc  hl
045B 56      ld  d,(hl)
045C 23      inc  hl
045D 08      ex  af,af'
045E 7E      ld  a,(hl)
045F FEFC    cp  FC          Rom# > 252 ?
0461 30BE    jr  nc,0421    oui
0463 06DF    ld  b,DF      activer Expansion Rom

```

KERNEL

0465	ED79	out	(c),a	
0467	21A8B1	ld	hl,B1A8	Rom ext. act.
046A	46	ld	b,(hl)	
046B	77	ld	(hl),a	
046C	C5	push	bc	
046D	FDE5	push	iy	
046F	3D	dec	a	
0470	FE07	cp	07	
0472	300F	jr	nc,0483	
0474	87	add	a,a	
0475	C6AC	add	a,AC	
0477	6F	ld	l,a	
0478	CEB1	adc	a,B1	
047A	95	sub	l	
047B	67	ld	h,a	
047C	7E	ld	a,(hl)	
047D	23	inc	hl	
047E	66	ld	h,(hl)	
047F	6F	ld	l,a	
0480	E5	push	hl	
0481	FDE1	pop	iy	
0483	067F	ld	b,7F	
0485	79	ld	a,c	
0486	CBD7	set	2,a	L Rom disable
0488	CB9F	res	3,a	U Rom enable
048A	CDA8B9	call	B9A8 (0439)	préparer config. et exécuter saut
048D	FDE1	pop	iy	
048F	F3	di		
0490	D9	exx		
0491	08	ex	af,af'	
0492	59	ld	e,c	
0493	C1	pop	bc	restaurer
0494	78	ld	a,b	ancienne
0495	06DF	ld	b,DF	configuration
0497	ED79	out	(c),a	de la Rom
0499	32A8B1	ld	(B1A8),a	(Rom ext. act.)
049C	067F	ld	b,7F	
049E	7B	ld	a,e	
049F	188F	jr	0430	

KERNEL

***** KL SIDE PCHL CONT'D

```

04A1 F3      di
04A2 E5      push hl
04A3 D9      exx
04A4 D1      pop  de
04A5 1808    jr   04AF
    
```

***** RST 2 LOW SIDE CALL

CONT'D0

```

04A7 F3      di
04A8 D9      exx
04A9 E1      pop  hl
04AA 5E      ld   e,(hl)
04AB 23      inc  hl
04AC 56      ld   d,(hl)
04AD 23      inc  hl
04AE E5      push hl
04AF 08      ex   af,af'
04B0 7A      ld   a,d
04B1 CBFA    set  7,d
04B3 CBF2    set  6,d
04B5 E6C0    and  C0
04B7 07      rlca
04B8 07      rlca
04B9 21ABB1  ld   hl,B1AB    config. Rom act.
04BC 86      add  a,(hl)
04BD 18A4    jr   0463
    
```

***** RST 5 FIRM JUMP CONT'D

```

04BF F3      di
04C0 D9      exx
04C1 E1      pop  hl
04C2 5E      ld   e,(hl)
04C3 23      inc  hl
04C4 56      ld   d,(hl)
04C5 CB91    res  2,c
04C7 ED49    out  (c),c      L Rom enable
04C9 ED533FBA ld  (BA3F),de    charger adr. de saut
04CD D9      exx
04CE FB      ei
    
```

KERNEL

```

04CF CD3EBA      call BA3E (04CF) et exécuter
04D2 F3         di
04D3 D9         exx
04D4 CBD1       set 2,c          L Rom disable
04D6 ED49       out (c),c
04D8 D9         exx
04D9 FB         ei
04DA C9         ret

```

***** KL L ROM ENABLE

```

04DB F3         di
04DC D9         exx
04DD 79         ld a,c
04DE CB91       res 2,c
04E0 ED49       out (c),c      L Rom enable
04E2 D9         exx
04E3 FB         ei
04E4 C9         ret

```

***** KL L ROM DISABLE

```

04E5 F3         di
04E6 D9         exx
04E7 79         ld a,c
04E8 CBD1       set 2,c
04EA ED49       out (c),c      L Rom disable
04EC D9         exx
04ED FB         ei
04EE C9         ret

```

***** KL U ROM ENABLE

```

04EF F3         di
04F0 D9         exx
04F1 79         ld a,c
04F2 CB99       res 3,c
04F4 ED49       out (c),c      U Rom enable
04F6 D9         exx
04F7 FB         ei
04F8 C9         ret

```


KERNEL

***** KL U ROM DISABLE

```

04F9 F3      di
04FA D9      exx
04FB 79      ld    a,c
04FC CBD9    set  3,c
04FE ED49    out  (c),c    U Rom disable
0500 D9      exx
0501 FB      ei
0502 C9      ret

```

***** KL ROM RESTORE

```

0503 F3      di
0504 D9      exx          a contient
0505 A9      xor    c          l'ancienne
0506 E60C    and   0C          configuration
0508 A9      xor    c
0509 4F      ld    c,a
050A ED49    out  (c),c
050C D9      exx
050D FB      ei
050E C9      ret

```

***** KL ROM SELECT

```

050F CD5EBA  call  BA5E (04EF) KL U ROM ENABLE CONT'D
0512 180F    jr    0523

```

***** KL PROBE ROM

```

0514 CD7EBA  call  BA7E (050F) KL ROM SELECT CONT'D
0517 3A00C0  ld    a,(C000)
051A 2A01C0  ld    hl,(C001)

```

***** KL ROM DESELECT

```

051D F5      push af
051E 78      ld    a,b
051F CD72BA  call  BA72 (0503) KL ROM RESTORE CONT'D
0522 F1      pop   af
0523 E5      push hl
0524 F3      di
0525 06DF    ld    b,DF    Expansion Rom (# dans c)
0527 ED49    out  (c),c    activer

```

KERNEL

```

0529 21A8B1      ld  hl,B1A8      Rom ext. act.
052C 46          ld  b,(hl)
052D 71          ld  (hl),c
052E 48          ld  c,b
052F 47          ld  b,a
0530 FB          ei
0531 E1          pop  hl
0532 C9          ret

```

***** KL CURR SELECTION

```

0533 3AA8B1      ld  a,(B1A8)     (Rom ext. act.)
0536 C9          ret

```

***** KL LDIR

```

0537 CDB2BA      call BAB2 (0543)
053A EDB0        ldir
053C C9          ret

```

***** KL LDDR

```

053D CDB2BA      call BAB2 (0543)
0540 EDB8        lddr
0542 C9          ret

```

***** Rom off & save config.

```

0543 F3          di
0544 D9          exx
0545 E1          pop  hl          manipuler adr. RET
0546 C5          push bc          rangeer ancienne config.
0547 CBD1        set  2,c          Roms
0549 CBD9        set  3,c          disable
054B ED49        out  (c),c
054D CDC7BA      call          call (hl)
0550 F3          di
0551 D9          exx          rétablir ancienne
0552 C1          pop  bc          configuration
0553 ED49        out  (c),c
0555 D9          exx
0556 FB          ei
0557 C9          ret

```

```

0558 E5          push hl          manipuler adr. RET

```

KERNEL

```
0559 D9      exx
055A FB      ei
055B C9      ret
```

***** RAM LAM

```
055C F3      di
055D D9      exx
055E 59      ld    e,c
055F CBD3    set  2,e    Roms
0561 CBDB    set  3,e    disable
0563 ED59    out  (c),e
0565 D9      exx
0566 7E      ld    a,(hl)  aller chercher octet
0567 D9      exx
0568 ED49    out  (c),c    fixer ancienne config.
056A D9      exx
056B FB      ei
056C C9      ret
```

***** RAM LAM (IX)

```
056D D9      exx
056E 79      ld    a,c    -
056F F60C    or    0C     Roms
0571 ED79    out  (c),a   disable
0573 DD7E00  ld    a,(ix+00) aller chercher octet
0576 ED49    out  (c),c   fixer ancienne config.
0578 D9      exx
0579 C9      ret

057A C7      rst  0
057B C7      rst  0
057C C7      rst  0
057D C7      rst  0
057E C7      rst  0
057F C7      rst  0
```

MACHINE PACK

2.5.2 MACHINE PACK (MC)

C'est la partie du système d'exploitation qui est la plus proche de la machine.

C'est ici que sont traités les divers interfaces et éléments périphériques tels que PIO et PSG. Cette procédure présente l'avantage qu'en cas de modification éventuelle de l'électronique, seul le MACHINE PACK devra être adapté comme par exemple le BIOS en CP/M.

De ce fait, seules quelques routines peuvent être utilisées souvent. Voici celles que nous avons sélectionnées:

MC PRINT CHAR sort le caractère qui se trouve dans a sur le port centronics. Après retour de la routine, le carry est mis si le caractère a bien été reçu.

MC SOUND REGISTER est intéressant pour les amateurs de musique. Sans que vous ne vous torturiez l'esprit avec la transmission de données au PSG qui est relativement compliquée, il vous suffit de transmettre le numéro de registre et l'octet souhaités en les plaçant respectivement dans a et c.

MACHINE PACK

***** RESET CONT'D

```

0580 F3          dl
0581 0182F7      ld  bc,F782   Control
0584 ED49       out  (c),c
0586 0100F4      ld  bc,F400   Port A
0589 ED49       out  (c),c
058B 0100F6      ld  bc,F600   Port C
058E ED49       out  (c),c
0590 017FEF      ld  bc,EF7F   Centronics
0593 ED49       out  (c),c
0595 06F5        ld  b,F5      Port B
0597 ED78        in  a,(c)
0599 E610        and  10        isoler LK4
059B 21C405      ld  hl,05C4   fin de table 60Hz
059E 2003        jr  nz,05A3  50Hz ? non =>
05A0 21D405      ld  hl,05D4   fin de table 50Hz
05A3 010FBC      ld  bc,BC0F
05A6 ED49       out  (c),c    charger adr. reg. video
05A8 2B          dec  hl
05A9 7E          ld  a,(hl)
05AA 04          inc  b
05AB ED79       out  (c),a    charger reg. video
05AD 05          dec  b
05AE 0D          dec  c
05AF F2A605      jp  p,05A6
05B2 1820        jr  05D4

```

***** Table 60Hz

```

05B4 3F 28 2E 8E 26 00 19 1E
05BC 00 07 00 00 30 00 C0 00

```

***** Table 50Hz

```

05C4 3F 28 2E 8E 1F 06 19 1B
05CC 00 07 00 00 30 00 C0 00

```

```

05D4 115C06      ld  de,065C   adresse de suite
05D7 210000      ld  hl,0000   reset
05DA 1832        jr  060E

```

***** MC BOOT PROGRAM

MACHINE PACK

```

05DC 310C0      ld    sp,C000
05DF E5         push  hl
05E0 CD681E     call  1E68      SOUND RESET
05E3 F3        di
05E4 01FFF8     ld    bc,F8FF   rétablir peripherie
05E7 ED49      out   (c),c
05E9 CD5C00     call  005C      KL CHOKE OFF
05EC E1         pop   hl
05ED D5         push  de
05EE C5         push  bc
05EF E5         push  hl
05F0 CD1E1A     call  1A1E      KM RESET
05F3 CD8810     call  1088      TXT RESET
05F6 CDB10A     call  0AB1      SCR RESET
05F9 CD5EBA     call  BASE (04EF) KL U ROM ENABLE CONT'D
05FC E1         pop   hl
05FD CD7507     call  0775      Jp (hl)
0600 C1         pop   bc
0601 D1         pop   de
0602 3807       Jr    c,060B    MC START PROGRAM
0604 EB         ex    de,hl
0605 48         ld    c,b
0606 11E806     ld    de,06E8   erreur de chargement
0609 1803       Jr    060E

```

***** MC START PROGRAM

```

060B 112607     ld    de,0726   rencontre RET après 0654
060E F3        di
060F ED56       im    1
0611 D9        exx
0612 0100DF     ld    bc,DF00   Palette Pointer reset
0615 ED49      out   (c),c
0617 01FFF8     ld    bc,F8FF   reset périphérie
061A ED49      out   (c),c     éventuellement connectée
061C 2100B1     ld    hl,B100   vider firmware-
061F 1101B1     ld    de,B101   Ram
0622 01FF07     ld    bc,07FF
0625 3600       ld    (hl),00
0627 EDB0       ldir
0629 01897F     ld    bc,7F89   U Rom off & L Rom on

```

MACHINE PACK

```

062C ED49      out   (c),c      Screen Mode 1
062E D9        exx
062F AF        xor   a
0630 08        ex    af,af'
0631 3100C0    ld    sp,C000
0634 E5        push hl
0635 C5        push bc
0636 D5        push de
0637 CD4400    call  0044      Restore High Kernel Jumps
063A CD8808    call  0888      JUMP RESTORE
063D CDE019    call  19E0      KM INITIALISE
0640 CD681E    call  1E68      SOUND RESET
0643 CDA00A    call  0AA0      SCR INITIALISE
0646 CD7810    call  1078      TXT INITIALISE
0649 CDB015    call  15B0      GRA INITIALISE
064C CD7023    call  2370      CAS INITIALISE
064F CDE607    call  07E6      MC RESET PRINTER
0652 FB        ei
0653 E1        pop   hl
0654 CD7507    call  0775      Jp (hl)
0657 C1        pop   bc
0658 E1        pop   hl
0659 C37700    jp    0077      U initialiser Rom

```

ç

***** Reset

```

065C CD1207    call  0712      sortir nom de société
065F CDEB06    call  06EB      sortir messages
0662 216D06    ld    hl,066D   message de mise sous tension
0665 CDEB06    call  06EB      sortir messages
0668 219306    ld    hl,0693   Copyright
066B 187E      jr    06EB      Sortir messages

```

***** Message de mise sous tension

```

066D 20 36 34 4B 20 4D 69 63      64K Mic
0675 72 6F 63 6F 6D 70 75 74      rocomput
067D 65 72 20 20 28 76 31 29      er (v1)
0685 0D 0A 0D 0A 00 43 6F 70      ....Cop
068D 79 72 69 67 68 74 20 A4      yright
0695 31 39 38 34 20 41 6D 73      19846Ams

```

MACHINE PACK

069D	74 72 61 64 20 43 6F 6E			trad Con
06A5	73 75 6D 65 72 20 45 6C			sumer El
06AD	65 63 74 72 6F 6E 69 63			ectronic
06B5	73 20 70 6C 63 0D 0A 20			s plc..
06BD	20 20 20 20 20 20 20 20			
06C5	20 20 61 6E 64 20 4C 6F			and Lo
06CD	63 6F 6D 6F 74 69 76 65			comotive
06D5	20 53 6F 66 74 77 61 72			Softwar
06DD	65 20 4C 74 64 2E 0D 0A			e Ltd...
06E5	0D 0A 00			...

06E8 21F406 ld hl,06F4 message erreur de chargement

***** Sortir messages

06EB	7E	ld	a,(hl)	
06EC	23	inc	hl	
06ED	B7	or	a	
06EE	C8	ret	z	
06EF	CD0014	call	1400	TXT OUTPUT
06F2	18F7	jr	06EB	Sortir messages

***** message erreur de chargement

06F4	2A 2A 2A 20 50 52 4F 47			*** PROG
06FC	52 41 4D 20 4C 4F 41 44			RAM LOAD
0704	20 46 41 49 4C 45 44 20			FAILED
070C	2A 2A 2A 0D 0A 00			***...

0712	06F5	ld	b,F5	
0714	ED78	in	a,(c)	Port B
0716	2F	cpl		
0717	E60E	and	OE	isoler LK1...3
0719	0F	rrca		/2
071A	212707	ld	hl,0727	nom de société.
071D	3C	inc	a	
071E	47	ld	b,a	
071F	7E	ld	a,(hl)	
0720	23	inc	hl	
0721	B7	or	a	
0722	20FB	jr	nz,071F	
0724	10F9	djnz	071F	

MACHINE PACK

0726 C9 ret

***** Noms de société

0727	41 72 6E 6F 6C 64 00 0A	Arnold..
072F	20 41 6D 73 74 72 61 64	Amstrad
0737	00 0A 20 4F 72 69 6F 6E	.. Orion
073F	00 0A 20 53 63 68 6E 65	.. Schne
0747	69 64 65 72 00 0A 20 41	ider.. A
074F	77 61 00 0A 20 53 6F 6C	wa.. Sol
0757	61 76 6F 78 00 0A 20 53	avox.. S
075F	61 69 73 68 6F 00 0A 20	aisho..
0767	54 72 69 75 6D 70 68 00	Triumph.
076F	0A 20 49 73 70 00	. Isp.

0775 E9 jp (h1)

***** MC SET MODE

0776	FE03	cp	03	Mode > 2 ?
0778	D0	ret	nc	Oui =>
0779	F3	d1		
077A	D9	exx		
077B	CB89	res	1,c	reset Mode Bits
077D	CB81	res	0,c	
077F	B1	or	c	
0780	4F	ld	c,a	fixer nouveau mode
0781	ED49	out	(c),c	
0783	FB	ei		
0784	D9	exx		
0785	C9	ret		

***** MC CLEAR INKS

0786	C5	push	bc	
0787	D5	push	de	
0788	01107F	ld	bc,7F10	
078B	CDAB07	call	07AB	Sortir couleur
078E	0E00	ld	c,00	
0790	CDAB07	call	07AB	Sortir couleur
0793	1B	dec	de	
0794	20FA	jr	nz,0790	
0796	D1	pop	de	

MACHINE PACK

0797 C1 pop bc
 0798 C9 ret

***** MC SET INKS

0799 C5 push bc
 079A D5 push de
 079B 01107F ld bc,7F10 couleur bord
 079E CDAB07 call 07AB Sortir couleur
 07A1 0E00 ld c,00 Adr. Ink 0
 07A3 CDAB07 call 07AB Sortir couleur
 07A6 20FB jr nz,07A3 charger toutes les
 07A8 D1 pop de mémoires couleurs
 07A9 C1 pop bc
 07AA C9 ret

***** Sortir couleur

07AB ED49 out (c),c Palette Pointer
 07AD 1A ld a,(de)
 07AE 13 inc de
 07AF E61F and 1F
 07B1 F640 or 40
 07B3 ED79 out (c),a couleur
 07B5 0C inc c
 07B6 79 ld a,c
 07B7 FE10 cp 10
 07B9 C9 ret

***** MC WAIT FLYBACK

07BA F5 push af
 07BB C5 push bc
 07BC 06F5 ld b,F5 Port B
 07BE ED78 in a,(c)
 07C0 1F rra VSYNC ?
 07C1 30FB jr nc,07BE non => attendre
 07C3 C1 pop bc
 07C4 F1 pop af
 07C5 C9 ret

***** MC SCREEN OFFSET

07C6 C5 push bc

MACHINE PACK

```

07C7 0F          rrca
07C8 0F          rrca
07C9 E630        and  30
07CB 4F          ld   c,a
07CC 7C          ld   a,h
07CD 1F          rra
07CE E603        and  03
07D0 B1          or   c
07D1 010CBC      ld   bc,BC0C
07D4 ED49        out  (c),c      Video Contr Reg 12
07D6 04          inc  b
07D7 ED79        out  (c),a      Début écran H1
07D9 05          dec  b
07DA 0C          inc  c
07DB ED49        out  (c),c      Reg 13
07DD 04          inc  b
07DE 7C          ld   a,h
07DF 1F          rra
07E0 7D          ld   a,l
07E1 1F          rra
07E2 ED79        out  (c),a      Début écran Lo
07E4 C1          pop  bc
07E5 C9          ret

```

***** MC RESET PRINTER

```

07E6 21EC07      ld   h1,07EC    Restore Printer Indirection
07E9 C38A0A      jp   0A8A       Move (h1+3)=>((h1+1)),cnt=(h1)

07EC 03          db   03         3 octets
07ED F1BD         dw   BDF1       adresse objet
07EF C3F807      jp   07F8       MC WAIT PRINTER

```

***** MC PRINT CHAR

```

07F2 C5          push bc
07F3 CDF1BD      call BDF1       MC WAIT PRINTER
07F6 C1          pop  bc
07F7 C9          ret

```

***** MC WAIT PRINTER

```

07F8 013200      ld   bc,0032

```

MACHINE PACK

```

07FB CD1B08      call 081B      MC BUSY PRINTER
07FE 3007        jr    nc,0807  MC SEND PRINTER
0800 10F9        djnz 07FB
0802 0D          dec    c
0803 20F6        jr    nz,07FB
0805 B7          or     a
0806 C9          ret

```

***** MC SEND PRINTER

```

0807 C5          push  bc
0808 06EF        ld    b,EF
080A E67F        and   7F      octet sans strobe
080C ED79        out   (c),a   vers l'imprimante
080E F680        or    80
0810 F3          di
0811 ED79        out   (c),a   Strobe mis
0813 E67F        and   7F
0815 FB          ei
0816 ED79        out   (c),a   Strobe éteint
0818 C1          pop   bc
0819 37          scf
081A C9          ret

```

***** MC BUSY PRINTER

```

081B C5          push  bc
081C 4F          ld    c,a
081D 06F5        ld    b,F5    Port B
081F ED78        in    a,(c)
0821 17          rla      Printer Busy
0822 17          rla      => Carry
0823 79          ld    a,c
0824 C1          pop   bc
0825 C9          ret

```

***** MC SOUND REGISTER

```

0826 F3          di
0827 06F4        ld    b,F4    Port A
0829 ED79        out   (c),a   Sound Reg#
082B 06F6        ld    b,F6    Port C
082D ED78        in    a,(c)   Sound Chip

```

MACHINE PACK

```

082F F6C0      or   C0      sur entrée
0831 ED79      out  (c),a   & Strobe mis
0833 E63F      and  3F
0835 ED79      out  (c),a   Strobe éteint
0837 06F4      ld   b,F4    Port A
0839 ED49      out  (c),c   Données sound
083B 06F6      ld   b,F6    Port C
083D 4F        ld   c,a
083E F680      or   80
0840 ED79      out  (c),a
0842 ED49      out  (c),c   latcher données
0844 FB        ei
0845 C9        ret

```

***** Scan Keyboard

```

0846 010EF4    ld   bc,F40E Port A
0849 ED49      out  (c),c   Sound Reg 14 (Keyb X Input)
084B 06F6      ld   b,F6    Port C
084D ED78      in   a,(c)
084F E630      and  30
0851 4F        ld   c,a
0852 F6C0      or   C0
0854 ED79      out  (c),a   Strobe mis
0856 ED49      out  (c),c   Strobe éteint
0858 04        inc  b
0859 3E92      ld   a,92    Port A&B=Input
085B ED79      out  (c),a   Control
085D C5        push bc
085E CBF1      set  6,c
0860 06F6      ld   b,F6    Port C
0862 ED49      out  (c),c   Keyb Y Outp & X Inp
0864 06F4      ld   b,F4    Port A
0866 ED78      in   a,(c)   Données (Keyb X Inp) => a
0868 46        ld   b,(hl)
0869 77        ld   (hl),a
086A A0        and  b
086B 2F        cpl
086C 12        ld   (de),a
086D 23 inc     hl
086E 13        inc  de

```

MACHINE PACK

086F	0C	inc	c	Keyb Y +1
0870	79	ld	a,c	
0871	E60F	and	0F	
0873	FE0A	cp	0A	tous canaux Y traités ?
0875	20E9	Jr	nz,0860	non => suivant
0877	C1	pop	bc	
0878	3E82	ld	a,82	Port A Output
087A	ED79	out	(c),a	Control
087C	05	dec	b	
087D	ED49	out	(c),c	Port C
087F	C9	ret		
0880	C7	rst	0	
0881	C7	rst	0	
0882	C7	rst	0	
0883	C7	rst	0	
0884	C7	rst	0	
0885	C7	rst	0	
0886	C7	rst	0	
0887	C7	rst	0	

JUMP RESTORE

2.5.3 JUMP RESTORE (JRE)

Ce pack sert uniquement à affecter à nouveau aux adresses MAIN JUMP leurs valeurs par défaut.

Pour les FIRM JUMPS, un RST1 est placé devant, pour les ARITHMETIK JUMPS, c'est un RST5.

Si vous pensez que vous avez modifié trop de vecteurs, tirez simplement la manette d'alarme en appelant JUMP RESTORE.

C'est également conseillé lorsque vous sortez d'un programme dans lequel vous avez généreusement offert au système d'exploitation vbs propres routines.

JUMP RESTORE

***** JUMP RESTORE

```

0888 11AC08      ld  de,08AC      Main Jump Adr.
088B 2100BB      ld  hl,BB00
088E 01CFBF      ld  bc,BFCF      Cnt => b , RST1 => c
0891 CD9708      call 0897
0894 01EF30      ld  bc,30EF      Cnt => b , RST5 => c
0897 71           ld  (hl),c
0898 23           inc  hl
0899 1A           ld  a,(de)
089A 77           ld  (hl),a
089B 13           inc  de
089C 23           inc  hl
089D EB         ex   de,hl
089E 79           ld  a,c
089F 2F         cpl
08A0 07         rlc
08A1 07         rlc
08A2 E680      and  80
08A4 B6         or   (hl)
08A5 EB         ex   de,hl
08A6 77         ld  (hl),a
08A7 13         inc  de
08A8 23         inc  hl
08A9 10EC      djnz 0897
08AB C9         ret

```

***** Main Jump Adr.

```

08AC E019      dw  19E0      KM INITIALISE
08AE 1E1A      dw  1A1E      KM RESET
08B0 3C1A      dw  1A3C      KM WAIT CHAR
08B2 421A      dw  1A42      KM READ CHAR
08B4 771A      dw  1A77      KM CHAR RETURN
08B6 BD1A      dw  1ABD      KM SET EXPAND
08B8 2E1B      dw  1B2E      KM GET EXPAND
08BA 7B1A      dw  1A7B      KM EXP BUFFER
08BC 561B      dw  1B56      KM WAIT KEY
08BE 5C1B      dw  1B5C      KM READ KEY
08C0 BD1C      dw  1CBD      KM TEST KEY
08C2 B31B      dw  1BB3      KM GET STATE
08C4 5C1C      dw  1C5C      KM GET JOYSTICK

```


JUMP RESTORE

08C6	521D	dw	1D52	KM SET TRANSLATE
08C8	3E1D	dw	1D3E	KM GET TRANSLATE
08CA	571D	dw	1D57	KM SET SHIFT
08CC	431D	dw	1D43	KM GET SHIFT
08CE	5C1D	dw	1D5C	KM SET CONTROL
08D0	481D	dw	1D48	KM GET CONTROL
08D2	AB1C	dw	1CAB	KM SET REPEAT
08D4	A61C	dw	1CA6	KM GET REPEAT
08D6	6D1C	dw	1C6D	KM SET DELAY
08D8	691C	dw	1C69	KM GET DELAY
08DA	711C	dw	1C71	KM ARM BREAK
08DC	821C	dw	1C82	KM DISARM BREAK
08DE	901C	dw	1C90	KM BREAK EVENT
08E0	7810	dw	1078	TXT INITIALISE
08E2	8810	dw	1088	TXT RESET
08E4	5114	dw	1451	TXT VDU ENABLE
08E6	4B14	dw	144B	TXT VDU DISABLE
08E8	0014	dw	1400	TXT OUTPUT
08EA	3413	dw	1334	TXT WR CHAR
08EC	AB13	dw	13AB	TXT RD CHAR
08EE	A713	dw	13A7	TXT SET GRAPHIC
08F0	0C12	dw	120C	TXT WIN ENABLE
08F2	5612	dw	1256	TXT GET WINDOW
08F4	4015	dw	1540	TXT CLEAR WINDOW
08F6	5E11	dw	115E	TXT SET COLUMN
08F8	6911	dw	1169	TXT SET ROW
08FA	7411	dw	1174	TXT SET CURSOR
08FC	8011	dw	1180	TXT GET CURSOR
08FE	8912	dw	1289	TXT CUR ENABLE
0900	9A12	dw	129A	TXT CUR DISABLE
0902	7912	dw	1279	TXT CUR ON
0904	8112	dw	1281	TXT CUR OFF
0906	CE11	dw	11CE	TXT VALIDATE
0908	6812	dw	1268	TXT PLACE/REMOVE CURSOR
090A	6812	dw	1268	TXT PLACE/REMOVE CURSOR
090C	A912	dw	12A9	TXT SET PEN
090E	BD12	dw	12BD	TXT GET PEN
0910	AE12	dw	12AE	TXT SET PAPER
0912	C312	dw	12C3	TXT GET PAPER
0914	C912	dw	12C9	TXT INVERSE

JUMP RESTORE

0916	7A13	dw	137A	TXT SET BACK
0918	8713	dw	1387	TXT GET BACK
091A	D312	dw	12D3	TXT GET MATRIX
091C	F112	dw	12F1	TXT SET MATRIX
091E	FD12	dw	12FD	TXT SET M TABLE
0920	2A13	dw	132A	TXT GET M TABLE
0922	CB14	dw	14CB	TXT GET CONTROLS
0924	E810	dw	10E8	TXT STR SELECT
0926	0711	dw	1107	TXT SWAP STREAMS
0928	B015	dw	15B0	GRA INITIALISE
092A	DF15	dw	15DF	GRA RESET
092C	F415	dw	15F4	GRA MOVE ABSOLUTE
092E	F115	dw	15F1	GRA MOVE RELATIVE
0930	FC15	dw	15FC	GRA ASK CURSOR
0932	0416	dw	1604	GRA SET ORIGIN
0934	1216	dw	1612	GRA GET ORIGIN
0936	3417	dw	1734	GRA WIN WIDTH
0938	7917	dw	1779	GRA WIN HEIGHT
093A	A617	dw	17A6	GRA GET W WIDTH
093C	BC17	dw	17BC	GRA GET W HEIGHT
093E	C517	dw	17C5	GRA CLEAR WINDOW
0940	F617	dw	17F6	GRA SET PEN
0942	0418	dw	1804	GRA GET PEN
0944	FD17	dw	17FD	GRA SET PAPER
0946	0A18	dw	180A	GRA GET PAPER
0948	1318	dw	1813	GRA PLOT ABSOLUTE
094A	1018	dw	1810	GRA PLOT RELATIVE
094C	2718	dw	1827	GRA TEST ABSOLUTE
094E	2418	dw	1824	GRA TEST RELATIVE
0950	3918	dw	1839	GRA LINE ABSOLUTE
0952	3618	dw	1836	GRA LINE RELATIVE
0954	4519	dw	1945	GRA WR CHAR
0956	A00A	dw	0AA0	SCR INITIALISE
0958	B10A	dw	0AB1	SCR RESET
095A	3C0B	dw	0B3C	SCR SET OFFSET
095C	450B	dw	0B45	SCR SET BASE
095E	500B	dw	0B50	SCR GET LOCATION
0960	CA0A	dw	0ACA	SCR SET MODE
0962	ECOA	dw	0AEC	SCR GET MODE
0964	F70A	dw	0AF7	SCR CLEAR

JUMP RESTORE

0966	570B	dw	0B57	SCR CHAR LIMITS
0968	640B	dw	0B64	SCR CHAR POSITION
096A	A90B	dw	0BA9	SCR DOT POSITION
096C	F90B	dw	0BF9	SCR NEXT BYTE
096E	050C	dw	0C05	SCR PREV BYTE
0970	130C	dw	0C13	SCR NEXT LINE
0972	2D0C	dw	0C2D	SCR PREV LINE
0974	860C	dw	0C86	SCR INK ENCODE
0976	A00C	dw	0CA0	SCR INK DECODE
0978	EC0C	dw	0CEC	SCR SET INK
097A	140D	dw	0D14	SCR GET INK
097C	F10C	dw	0CF1	SCR SET BORDER
097E	190D	dw	0D19	SCR GET BORDER
0980	E40C	dw	0CE4	SCR SET FLASHING
0982	E80C	dw	0CE8	SCR GET FLASHING
0984	B30D	dw	0DB3	SCR FILL BOX
0986	B70D	dw	0DB7	SCR FLOOD BOX
0988	DF0D	dw	0DDF	SCR CHAR INVERT
098A	FA0D	dw	0DFA	SCR HW ROLL
098C	3E0E	dw	0E3E	SCR SW ROLL
098E	F30E	dw	0EF3	SCR UNPACK
0990	490F	dw	0F49	SCR REPACK
0992	490C	dw	0C49	SCR ACCESS
0994	6B0C	dw	0C6B	SCR PIXELS
0996	C40F	dw	0FC4	SCR HORIZONTAL
0998	2F10	dw	102F	SCR VERTICAL
099A	7023	dw	2370	CAS INITIALISE
099C	7F23	dw	237F	CAS SET SPEED
099E	8E23	dw	238E	CAS NOISY
09A0	4B2A	dw	2A4B	CAS START MOTOR
09A2	4F2A	dw	2A4F	CAS STOP MOTOR
09A4	512A	dw	2A51	CAS RESTORE MOTOR
09A6	9223	dw	2392	CAS IN OPEN
09A8	FC23	dw	23FC	CAS IN CLOSE
09AA	0124	dw	2401	CAS IN ABANDON
09AC	3524	dw	2435	CAS IN CHAR
09AE	AB24	dw	24AB	CAS IN DIRECT
09B0	9A24	dw	249A	CAS RETURN
09B2	9624	dw	2496	CAS TEST EOF
09B4	AB23	dw	23AB	CAS OUT OPEN

JUMP RESTORE

09B6	1524	dw	2415	CAS OUT CLOSE
09B8	2E24	dw	242E	CAS OUT ABANDON
09BA	5B24	dw	245B	CAS OUT CHAR
09BC	EA24	dw	24EA	CAS OUT DIRECT
09BE	2825	dw	2528	CAS CATALOG
09C0	3F28	dw	283F	CAS WRITE
09C2	3628	dw	2836	CAS READ
09C4	5128	dw	2851	CAS CHECK
09C6	681E	dw	1E68	SOUND RESET
09C8	9F1F	dw	1F9F	SOUND QUEUE
09CA	6C20	dw	206C	SOUND CHECK
09CC	8920	dw	2089	SOUND ARM EVENT
09CE	4A20	dw	204A	SOUND RELEASE
09D0	CB1E	dw	1ECB	SOUND HOLD
09D2	E61E	dw	1EE6	SOUND CONTINUE
09D4	3823	dw	2338	SOUND AMPL ENVELOPE
09D6	3D23	dw	233D	SOUND TONE ENVELOPE
09D8	4923	dw	2349	SOUND A ADDRESS
09DA	4E23	dw	234E	SOUND T ADDRESS
09DC	5C00	dw	005C	KL CHOKE OFF
09DE	2903	dw	0329	KL ROM WALK
09E0	3203	dw	0332	KL INIT BACK
09E2	A102	dw	02A1	KL LOG EXT
09E4	B202	dw	02B2	KL FIND COMMAND
09E6	6301	dw	0163	KL NEW FRAME FLY
09E8	6A01	dw	016A	KL ADD FRAME FLY
09EA	7001	dw	0170	KL DEL FRAME FLY
09EC	7601	dw	0176	KL NEW FAST TICKER
09EE	7D01	dw	017D	KL ADD FAST TICKER
09F0	8301	dw	0183	Delete Fast Ticker
09F2	B301	dw	01B3	KL ADD TICKER
09F4	C501	dw	01C5	Delete Ticker
09F6	D201	dw	01D2	KL INIT EVENT
09F8	E201	dw	01E2	KL EVENT
09FA	2802	dw	0228	KL SYNC RESET
09FC	8502	dw	0285	KL DEL SYNCHRONOUS
09FE	5602	dw	0256	KL NEXT SYNC
0A00	1A02	dw	021A	KL DO SYNC
0A02	7702	dw	0277	KL DONE SYNC
0A04	9502	dw	0295	KL EVENT DISABLE

JUMP RESTORE

0A06	9B02	dw	029B	KL EVENT ENABLE
0A08	8E02	dw	028E	KL DISARM EVENT
0A0A	9900	dw	0099	KL TIME PLEASE
0A0C	A300	dw	00A3	KL TIME SET
0A0E	DC05	dw	05DC	MC BOOT PROGRAM
0A10	0B06	dw	060B	MC START PROGRAM
0A12	BA07	dw	07BA	MC WAIT FLYBACK
0A14	7607	dw	0776	MC SET MODE
0A16	C607	dw	07C6	MC SCREEN OFFSET
0A18	8607	dw	0786	MC CLEAR INKS
0A1A	9907	dw	0799	MC SET INKS
0A1C	E607	dw	07E6	MC RESET PRINTER
0A1E	F207	dw	07F2	MC PRINT CHAR
0A20	1B08	dw	081B	MC BUSY PRINTER
0A22	0708	dw	0807	MC SEND PRINTER
0A24	2608	dw	0826	MC SOUND REGISTER
0A26	8808	dw	0888	JUMP RESTORE

***** Basic Jump Adr.

0A28	982A	dw	2A98	EDIT
0A2A	182E	dw	2E18	FLO Var. (de) => (hl)
0A2C	292E	dw	2E29	FLO Int => Flo
0A2E	552E	dw	2E55	FLO 4-octets => Flo
0A30	662E	dw	2E66	FLO Flo => Int
0A32	8E2E	dw	2E8E	FLO Flo => Int
0A34	A12E	dw	2EA1	FLO Fix
0A36	AC2E	dw	2EAC	FLO Int
0A38	B62E	dw	2EB6	
0A3A	1D2F	dw	2F1D	FLO nombre * 10 ^a
0A3C	3F33	dw	333F	FLO Add
0A3E	3733	dw	3337	FLO Sub
0A40	3B33	dw	333B	FLO Sub
0A42	1534	dw	3415	FLO Mul
0A44	9E34	dw	349E	FLO Div
0A46	7835	dw	3578	FLO nombre * 2 ^a
0A48	9A35	dw	359A	FLO Cmp
0A4A	F835	dw	35F8	FLO +/-
0A4C	E835	dw	35E8	FLO Sgn
0A4E	AE31	dw	31AE	FLO Deg/Rad
0A50	A331	dw	31A3	FLO Pi

JUMP RESTORE

0A52	0A31	dw	310A	FLO Racine
0A54	0D31	dw	310D	FLO Puissance
0A56	1430	dw	3014	FLO Log
0A58	0F30	dw	300F	FLO Log10
0A5A	9030	dw	3090	FLO Exp
0A5C	BC31	dw	31BC	FLO Sin
0A5E	B231	dw	31B2	FLO Cos
0A60	3132	dw	3231	FLO Tan
0A62	4132	dw	3241	FLO Atn
0A64	5E2E	dw	2E5E	FLO 4-Octets * 256 => Flo
0A66	942F	dw	2F94	FLO RND Init
0A68	A12F	dw	2FA1	FLO SET RANDOM SEED
0A6A	B72F	dw	2FB7	FLO RND
0A6C	E62F	dw	2FE6	FLO GET LAST RND
0A6E	0837	dw	3708	
0A70	0E37	dw	370E	
0A72	1537	dw	3715	INT mettre Sgn dans b
0A74	2837	dw	3728	INT Add
0A76	3137	dw	3731	INT Sub
0A78	3037	dw	3730	INT Sub
0A7A	3937	dw	3739	INT Mul
0A7C	7A37	dw	377A	INT Div
0A7E	8137	dw	3781	INT Mod
0A80	5037	dw	3750	INT Mul unsigned
0A82	8C37	dw	378C	h1/de => h1, Rest => de
0A84	E937	dw	37E9	INT Cmp
0A86	D437	dw	37D4	INT +/-
0A88	E037	dw	37E0	INT Sgn

***** Move (h1+3)=>((h1+1)),cnt=(h1)0

0A8A	4E	ld	c, (h1)
0A8B	0600	ld	b,00
0A8D	23	inc	h1
0A8E	5E	ld	e, (h1)
0A8F	23	inc	h1
0A90	56	ld	d, (h1)
0A91	23	inc	h1
0A92	EDB0	ldir	
0A94	C9	ret	

JUMP RESTORE

0A95	C7	rst	0
0A96	C7	rst	0
0A97	C7	rst	0
0A98	C7	rst	0
0A99	C7	rst	0
0A9A	C7	rst	0
0A9B	C7	rst	0
0A9C	C7	rst	0
0A9D	C7	rst	0
0A9E	C7	rst	0
0A9F	C7	rst	0

SCREEN PACK

2.5.4 SCREEN PACK (SCR)

Le SCREEN PACK est subordonné au TEXT PACK et au GRAPHICS PACK. Il se charge de la réalisation pratique des tâches ordonnées par ces deux packs. Il est en effet responsable du traitement direct de l'écran.

Voici les routines que nous voulons vous présenter :

SCR NEXT BYTE et SCR PREV BYTE fournissent dans h1 l'adresse écran de la prochaine ou de la dernière position d'octet, lorsque vous placez dans h1, avant d'appeler la routine, l'ancienne adresse. C'est aussi pratique que cela semble superflu. En effet, du fait de l'organisation de l'écran, il n'est pas facile de déterminer la position d'octet. La distance dépend en outre du mode. Notez que si la prochaine ou la dernière position sort du cadre de l'écran, l'adresse fournie en retour n'a pas de sens. Elles se trouve en effet alors dans la zone des 48 derniers octets de la Ram vidéo, qui ne sont pas utilisés pour la représentation sur l'écran.

SCR NEXT LINE et SCR PREV LINE travaillent de façon similaire, si ce n'est que l'adresse écran est calculée une ligne entière avant ou après. Ici également, l'adresse n'a pas de signification lorsqu'on sort de la zone représentable.

SCR HW ROLL décale l'écran d'une ligne vers le bas lorsque b=0 et d'une ligne vers le haut lorsque b<>0. a doit recevoir la couleur que devra avoir la nouvelle ligne (vide) qui sera ajoutée.

SCR SW ROLL décale une zone de l'écran. a et b doivent être servis comme ci-dessus. h doit en outre recevoir le numéro de colonne du bord gauche de la zone à décaler, l la ligne supérieure, d la colonne droite et e la ligne inférieure de cette zone.

Notez que colonne et ligne 0 correspondent à l'angle supérieur gauche de l'écran. Faites vous-même très attention à ce que les paramètres transmis marquent bien une zone comprise dans la Ram vidéo.

SCREEN PACK

***** SCR INITIALISE

```
OAAO 114D10      ld  de,104D      Couleurs par défaut
OAA3 CD8607      call 0786        MC CLEAR INKS
OAA6 3ECO        ld  a,C0
OAA8 32CBB1      ld  (B1CB),a     (High Byte début écran)
OAA8 32CBB1      ld  (B1CB),a     (High Byte début écran)
OAB  CDB10A      call 0AB1        SCR RESET
OAAE  C3F20A      jp  OAF2
```

***** SCR RESET

```
OAB1 AF          xor  a
OAB2 CD490C      call 0C49        SCR ACCESS
OAB5 21BEOA      ld  hl,0ABE      Restore SCR Indirections
OAB8 CD8A0A      call 0A8A        Move (hl+3)=>((hl+1)),cnt=(hl)
OABB C3D20C      jp  OCD2        Reset couleurs

OABE 09          db  09          9 octets
OABF E5BD        dw  BDE5        Adresse objet
OAC1 C3820C      jp  OC82        SCR READ

OAC4 C3680C      jp  OC68        SCR WRITE

OAC7 C3F70A      jp  OAF7        SCR CLEAR
```

***** SCR SET MODE

```
OACA E603        and  03
OACC FE03        cp   03
OACE D0          ret  nc
OACF F5          push af
OADO CD4F0D      call 0D4F
OAD3 F1          pop  af
OAD4 5F          ld  e,a
OAD5 CDB710      call 10B7
OAD8 F5          push af
OAD9 CDD615      call 15D6
OADC E5          push hl
OADD 7B          ld  a,e
OADE CD110B      call 0B11        charger cartes bits
OAE1 CDEBBD      call BDEB        SCR MODE CLEAR
OAE4 E1          pop  hl
OAE5 CDB615      call 15B6
```

SCREEN PACK

OAE8 F1 pop af
 OAE9 C3D510 jp 10D5

***** SCR GET MODE

OAE8 3AC8B1 ld a,(B1C8) (curr. Screen Mode)
 OAE9 FE01 cp 01
 OAF1 C9 ret

OAF2 3E01 ld a,01
 OAF4 CD110B call OB11 Charger cartes bits

ç

***** SCR MODE CLEAR

OAF7 CD4F0D call OD4F
 OAF8 210000 ld hl,0000
 OAFD CD3C0B call OB3C SCR SET OFFSET
 OB00 2ACAB1 ld hl,(B1CA) (Adr. Screen Start)
 OB03 2E00 ld l,00
 OB05 54 ld d,h hl=adresse de base
 OB06 1E01 ld e,01 de=adresse de base +1
 OB08 01FF3F ld bc,3FFF 16k
 OB0B 75 ld (hl),l
 OB0C EDB0 ldir vider l'écran
 OB0E C33C0D jp OD3C

***** Charger cartes bits

OB11 213A0B ld hl,OB3A Cartes bits Mode 0
 OB14 FE01 cp 01
 OB16 3808 jr c,OB20
 OB18 21360B ld hl,OB36 Cartes bits Mode 1
 OB1B 2803 jr z,OB20
 OB1D 212E0B ld hl,OB2E Cartes bits Mode 2
 OB20 11CFB1 ld de,B1CF Cartes bits suivant Mode
 OB23 010800 ld bc,0008
 OB26 EDB0 ldir
 OB28 32C8B1 ld (B1C8),a (curr. Screen Mode)
 OB2B C37607 jp 0776 MC SET MODE

***** Cartes bits Mode 2

OB2E 80 40 20 10 08 04 02 01

SCREEN PACK

***** Cartes bits Mode 1
 OB36 88 44 22 11

***** Cartes bits Mode 0
 OB3A AA 55

***** SCR SET OFFSET

OB3C 7C ld a,h
 OB3D E607 and 07
 OB3F 67 ld h,a
 OB40 22C9B1 ld (B1C9),h1
 OB43 1805 jr OB4A

***** SCR SET BASE

OB45 E6C0 and C0
 OB47 32CBB1 ld (B1CB),a (High Byte Screen Start)
 OB4A CD500B call OB50 SCR GET LOCATION
 OB4D C3C607 jp 07C6 MC SCREEN OFFSET

***** SCR GET LOCATION

OB50 2AC9B1 ld h1,(B1C9)
 OB53 3ACBB1 ld a,(B1CB) (High Byte Screen Start)
 OB56 C9 ret

ç

***** SCR CHAR LIMITS

OB57 CDECOA call 0AEC SCR GET MODE
 OB5A 011813 ld bc,1318
 OB5D D8 ret c
 OB5E 0627 ld b,27
 OB60 C8 ret z
 OB61 064F ld b,4F
 OB63 C9 ret

***** SCR CHAR POSITION

OB64 D5 push de
 OB65 CDECOA call 0AEC SCR GET MODE
 OB68 0604 ld b,04
 OB6A 3805 jr c,OB71
 OB6C 0602 ld b,02

SCREEN PACK

0B6E	2801	jr	z,0B71	
0B70	05	dec	b	
0B71	C5	push	bc	
0B72	5C	ld	e,h	
0B73	1600	ld	d,00	
0B75	62	ld	h,d	
0B76	D5	push	de	
0B77	54	ld	d,h	
0B78	5D	ld	e,l	
0B79	29	add	hl,hl	
0B7A	29	add	hl,hl	
0B7B	19	add	hl,de	
0B7C	29	add	hl,hl	
0B7D	29	add	hl,hl	
0B7E	29	add	hl,hl	
0B7F	29	add	hl,hl	
0B80	D1	pop	de	
0B81	19	add	hl,de	
0B82	10FD	djnz	0B81	
0B84	ED5BC9B1	ld	de,(B1C9)	
0B88	19	add	hl,de	
0B89	7C	ld	a,h	
0B8A	E607	and	07	
0B8C	67	ld	h,a	
0B8D	3ACBB1	ld	a,(B1CB)	(High Byte Screen Start)
0B90	84	add	a,h	
0B91	67	ld	h,a	
0B92	C1	pop	bc	
0B93	D1	pop	de	
0B94	C9	ret		
0B95	7B	ld	a,e	
0B96	95	sub	l	
0B97	3C	inc	a	
0B98	87	add	a,a	
0B99	87	add	a,a	
0B9A	87	add	a,a	
0B9B	5F	ld	e,a	
0B9C	7A	ld	a,d	
0B9D	94	sub	h	

SCREEN PACK

```

OB9E 3C          inc  a
OB9F 57          ld   d,a
OBA0 CD640B     call OB64          SCR CHAR POSITION
OBA3 AF         xor  a
OBA4 82         add  a,d
OBA5 10FD      djnz OBA4
OBA7 57         ld   d,a
OBA8 C9        ret

```

***** SCR DOT POSITION

```

OBA9 D5        push de
OBAA EB        ex   de,h1
OBAB 21C700    ld   h1,00C7
OBAE B7        or   a
OBAF ED52     sbc  h1,de
OBB1 7D        ld   a,l
OBB2 E607     and  07
OBB4 87        add  a,a
OBB5 87        add  a,a
OBB6 87        add  a,a
OBB7 4F        ld   c,a
OBB8 7D        ld   a,l
OBB9 E6F8     and  F8
OB BB 6F       ld   l,a
OB BC 54       ld   d,h
OB BD 5D       ld   e,l
OB BE 29       add  h1,h1
OB BF 29       add  h1,h1
OB C0 19       add  h1,de
OB C1 29       add  h1,h1
OB C2 D1       pop  de
OB C3 CDECOA   call OAEC          SCR GET MODE
OB C6 0601     ld   b,01
OB C8 3806     jr   c,OBDO
OB CA 0603     ld   b,03
OB CC 2802     jr   z,OBDO
OB CE 0607     ld   b,07
OB DO 78       ld   a,b
OB D1 A3       and  e
OB D2 F5       push af

```

SCREEN PACK

```

OBD3 78          ld  a,b
OBD4 0F          rrca
OBD5 CB3A       srl  d
OBD7 CB1B       rr  e
OBD9 0F          rrca
OBDA 38F9       jr   c,OBD5
OBDC 19          add hl,de
OBDD ED5BC9B1   ld  de,(B1C9)
OBE1 19          add hl,de
OBE2 7C          ld  a,h
OBE3 E607       and  07
OBE5 67          ld  h,a
OBE6 3ACBB1     ld  a,(B1CB)   (High Byte Screen Start)
OBE9 84          add  a,h
OBEA 81          add  a,c
OBEB 67          ld  h,a
OBEC F1          pop  af
OBED E5          push hl
OBEE 1600       ld  d,00
OBF0 5F          ld  e,a
OBF1 21CFB1     ld  hl,B1CF   Cartes bits suivant Mode
OBF4 19          add  hl,de
OBF5 4E          ld  c,(hl)
OBF6 EB          ex  de,hl
OBF7 E1          pop  hl
OBF8 C9          ret

```

***** SCR NEXT BYTE

```

OBF9 2C          inc  l
OBFa C0          ret  nz
OBFb 24          inc  h
OBFc 7C          ld  a,h
OBFd E607       and  07
OBFf C0          ret  nz
OC00 7C          ld  a,h
OC01 D608       sub  08
OC03 67          ld  h,a
OC04 C9          ret

```

***** SCR PREV BYTE

SCREEN PACK

0C05 7D ld a,l
0C06 2D dec l
0C07 B7 or a
0C08 C0 ret nz
0C09 7C ld a,h
0C0A 25 dec h
0C0B E607 and 07
0C0D C0 ret nz
0C0E 7C ld a,h
0C0F C608 add a,08
0C11 67 ld h,a
0C12 C9 ret

***** SCR NEXT LINE

0C13 7C ld a,h
0C14 C608 add a,08
0C16 67 ld h,a
0C17 E638 and 38
0C19 C0 ret nz
0C1A 7C ld a,h
0C1B D640 sub 40
0C1D 67 ld h,a
0C1E 7D ld a,l
0C1F C650 add a,50
0C21 6F ld l,a
0C22 D0 ret nc
0C23 24 inc h
0C24 7C ld a,h
0C25 E607 and 07
0C27 C0 ret nz
0C28 7C ld a,h
0C29 D608 sub 08
0C2B 67 ld h,a
0C2C C9 ret

***** SCR PREV LINE

0C2D 7C ld a,h
0C2E D608 sub 08
0C30 67 ld h,a
0C31 E638 and 38

SCREEN PACK

```

0C33 FE38      cp    38
0C35 C0        ret   nz
0C36 7C        ld    a,h
0C37 C640      add   a,40
0C39 67        ld    h,a
0C3A 7D        ld    a,l
0C3B D650      sub   50
0C3D 6F        ld    l,a
0C3E D0        ret   nc
0C3F 7C        ld    a,h
0C40 25        dec   h
0C41 E607      and   07
0C43 C0        ret   nz
0C44 7C        ld    a,h
0C45 C608      add   a,08
0C47 67        ld    h,a
0C48 C9        ret

```

***** SCR ACCESS

```

0C49 E603      and   03
0C4B 216B0C    ld    h1,0C6B    SCR PIXELS (FORCE Mode)
0C4E 280F      jr    z,0C5F
0C50 FE02      cp    02
0C52 21720C    ld    h1,0C72    XOR Mode
0C55 3808      jr    c,0C5F
0C57 21770C    ld    h1,0C77    AND Mode
0C5A 2803      jr    z,0C5F
0C5C 217D0C    ld    h1,0C7D    OR Mode
0C5F 3EC3      ld    a,C3      Jp
0C61 32CCB1    ld    (B1CC),a  (SCR Write Indirection)
0C64 22CDB1    ld    (B1CD),h1
0C67 C9        ret

```

***** SCR WRITE

```

0C68 C3CCB1      jp    B1CC      SCR Write Indirection

```

***** SCR PIXELS (FORCE Mode)

```

0C6B 7E        ld    a,(h1)
0C6C A8        xor   b
0C6D B1        or    c

```


SCREEN PACK

```
0C6E A9      xor  c
0C6F A8      xor  b
0C70 77      ld   (hl),a
0C71 C9      ret
```

***** XOR Mode

```
0C72 78      ld   a,b
0C73 A1      and  c
0C74 AE      xor  (hl)
0C75 77      ld   (hl),a
0C76 C9      ret
```

***** AND Mode

```
0C77 79      ld   a,c
0C78 2F      cpl
0C79 B0      or   b
0C7A A6      and  (hl)
0C7B 77      ld   (hl),a
0C7C C9      ret
```

***** OR Mode

```
0C7D 78      ld   a,b
0C7E A1      and  c
0C7F B6      or   (hl)
0C80 77      ld   (hl),a
0C81 C9      ret
```

***** SCR READ

```
0C82 7E      ld   a,(hl)
0C83 C3AC0C  jp   0CAC
```

***** SCR INK ENCODE

```
0C86 C5      push bc
0C87 D5      push de
0C88 CDC20C  call 0CC2
0C8B 5F      ld   e,a
0C8C 0608    ld   b,08
0C8E 3ACFB1  ld   a,(B1CF)    (Cartes bits suivant Mode)
0C91 4F      ld   c,a
0C92 CB0B    rrc  e
```

SCREEN PACK

```

OC94 17          rla
OC95 CB09        rrc  c
OC97 3802        jr   c,OC9B
OC99 CB03        rlc  e
OC9B 10F5        djnz OC92
OC9D D1          pop  de
OC9E C1          pop  bc
OC9F C9          ret

```

***** SCR INK DECODE

```

OCA0 C5          push bc
OCA1 47          ld   b,a
OCA2 3ACFB1      ld   a,(B1CF)  (Cartes bits suivant Mode)
OCA5 4F          ld   c,a
OCA6 78          ld   a,b
OCA7 CDACOC      call OCAC
OCA8 C1          pop  bc
OCA9 C9          ret

```

```

OCAC D5          push de
OCAD 110800      ld   de,0008
OCB0 0F          rrca
OCB1 CB12        rl   d
OCB3 CB09        rrc  c
OCB5 3802        jr   c,OCB9
OCB7 CB1A        rr   d
OCB9 1D          dec  e
OCBA 20F4        jr   nz,OCB0
OCBC 7A          ld   a,d
OCBD CDC20C      call OCC2
OCC0 D1          pop  de
OCC1 C9          ret

```

```

OCC2 57          ld   d,a
OCC3 CDECOA      call OAEC  SCR GET MODE
OCC6 7A          ld   a,d
OCC7 D0          ret  nc
OCC8 0F          rrca
OCC9 0F          rrca
OCCA CE00        adc  a,00

```

SCREEN PACK

```
OCCC 0F          rrca
OCCD 9F          sbc  a,a
OCCE E606        and  06
OCD0 AA          xor  d
OCD1 C9          ret
```

***** Reset couleurs

```
OCD2 214D10      ld   hl,104D   Couleurs par défaut
OCD5 11D9B1      ld   de,B1D9   mémoire couleur 2èmes couleurs
OCD8 012200      ld   bc,0022
OCCB EDB0        ldir
OCCD AF          xor  a
OCDE 32FBB1      ld   (B1FB),a   (Flag jeu de couleurs act.)
OCE1 210A0A      ld   hl,0A0A   (0A0A)=9900
```

***** SCR SET FLASHING

```
OCE4 22D7B1      ld   (B1D7),hl   (Flash Periods)
OCE7 C9          ret
```

***** SCR GET FLASHING

```
OCE8 2AD7B1      ld   hl,(B1D7)   (Flash Periods)
OCEB C9          ret
```

***** SCR SET INK

```
OCEC E60F        and  0F
OCEE 3C          inc  a
OCEF 1801        jr   OCF2       Set Colour
```

***** SCR SET BORDER

```
OCF1 AF          xor  a
```

***** Set Colour

```
OCF2 5F          ld   e,a
OCF3 78          ld   a,b
OCF4 CDOA0D      call ODOA       Aller chercher entrée matrice
                                                couleurs
OCF7 46          ld   b,(hl)
OCF8 79          ld   a,c
OCF9 CDOA0D      call ODOA       Aller chercher entrée matrice
                                                couleurs
```

SCREEN PACK

```

OCFC 4E          ld  c,(hl)
OCFD 7B          ld  a,e
OCFE CD2F0D     call OD2F      aller chercher adr. Ink
OD01 71          ld  (hl),c
OD02 EB          ex  de,hl
OD03 70          ld  (hl),b
OD04 3EFF       ld  a,FF
OD06 32FCB1     ld  (B1FC),a
OD09 C9         ret

```

***** Aller chercher entrée matrice couleurs

```

OD0A E61F       and  1F
OD0C C693       add  a,93
OD0E 6F         ld  l,a
OD0F CE0D       adc  a,0D
OD11 95         sub  l
OD12 67         ld  h,a
OD13 C9         ret

```

***** SCR GET INK

```

OD14 E60F       and  0F
OD16 3C         inc  a
OD17 1801       jr   OD1A      Get Colour

```

***** SCR GET BORDER

```

OD19 AF         xor  a

```

***** Get Colour

```

OD1A CD2F0D     call OD2F      aller chercher adr. Ink
OD1D 1A         ld  a,(de)
OD1E 5E         ld  e,(hl)
OD1F CD240D     call OD24
OD22 41         ld  b,c
OD23 7B         ld  a,e
OD24 0E00       ld  c,00
OD26 21930D     ld  hl,0D93    Matrice couleur
OD29 BE         cp  (hl)
OD2A C8         ret  z
OD2B 23         inc hl
OD2C 0C         inc  c

```

SCREEN PACK

```

OD2D 18FA      Jr  OD29
ç
***** aller chercher adr. Ink
OD2F 5F        ld  e,a
OD30 1600      ld  d,00
OD32 21EAB1    ld  hl,B1EA  mémoire couleur 1ères couleurs
OD35 19        add hl,de
OD36 EB        ex  de,hl
OD37 21EFFF    ld  hl,FFEF
OD3A 19        add hl,de
OD3B C9        ret

OD3C 21FEB1    ld  hl,B1FE  Event Block: Set Inks
OD3F E5        push hl
OD40 CD7001    call 0170  KL DEL FRAME FLY
OD43 CD6D0D    call OD6D  Flash Inks
OD46 115B0D    ld  de,OD5B  Set Inks on Frame Fly
OD49 0681      ld  b,81
OD4B E1        pop  hl
OD4C C36301    jp   0163  KL NEW FRAME FLY

OD4F 21FEB1    ld  hl,B1FE  Event Block: Set Inks
OD52 CD7001    call 0170  KL DEL FRAME FLY
OD55 CD810D    call OD81  aller chercher params de jeu
de couleurs actuel
OD58 C38607    jp   0786  MC CLEAR INKS

***** Set Inks on Frame Fly
OD5B 21FDB1    ld  hl,B1FD  curr. Flash Period
OD5E 35        dec  (hl)
OD5F 280C      Jr  z,OD6D  Flash Inks
OD61 2B        dec  hl
OD62 7E        ld  a,(hl)
OD63 B7        or   a
OD64 C8        ret  z
OD65 CD810D    call OD81  aller chercher params de jeu
de couleurs actuel
OD68 CD9907    call 0799  MC SET INKS
OD6B 180F      Jr  OD7C

```

SCREEN PACK

***** Flash Inks

```

OD6D CD810D      call  OD81      aller chercher params de jeu
                  de couleurs actuel
OD70 32FDB1      ld    (B1FD),a  (curr. Flash Period)
OD73 CD9907      call  0799      MC SET INKS
OD76 21FBB1      ld    hl,B1FB   Flag Jeu de couleurs act.
OD79 7E          ld    a,(hl)
OD7A 2F          cpl
OD7B 77          ld    (hl),a
OD7C AF          xor    a
OD7D 32FCB1      ld    (B1FC),a
OD80 C9          ret

```

***** aller chercher params de jeu de couleurs actuel

```

OD81 11EAB1      ld    de,B1EA   mémoire couleurs 1ères couleurs
OD84 3AFBB1      ld    a,(B1FB)  (Flag Jeu de couleurs act.)
OD87 B7          or    a
OD88 3AD8B1      ld    a,(B1D8)  (Flash Period 1.Colour)
OD8B C8          ret    z
OD8C 11D9B1      ld    de,B1D9   Mémoire couleurs 2èmes couleurs
OD8F 3AD7B1      ld    a,(B1D7)  (Flash Periods)
OD92 C9          ret

```

***** Matrice de couleurs

```

OD93 14 04 15 1C 18 1D 0C 05
OD9B 0D 16 06 17 1E 00 1F 0E
ODA3 07 0F 12 02 13 1A 19 1B
ODAB 0A 03 0B 01 08 09 10 11

```

***** SCR FILL BOX

```

ODB3 4F          ld    c,a
ODB4 CD950B      call  0B95

```

***** SCR FLOOD BOX

```

ODB7 E5          push hl
ODB8 7          A
                  ld    a,d
ODB9 CDE80E      call  0EE8
ODBC 3009        jr    nc,ODC7
ODBE 42          ld    b,d

```

SCREEN PACK

```

ODBF 71          ld  (h1),c
ODCO CDF90B     call 0BF9          SCR NEXT BYTE
ODC3 10FA       djnz 0DBF
ODC5 1810       jr   0DD7
ODC7 C5         push bc
ODC8 D5         push de
ODC9 71         ld  (h1),c
ODCA 15         dec  d
ODCB 2808       jr   z,0DD5
ODCD 4A         ld  c,d
ODCE 0600       ld  b,00
ODD0 54         ld  d,h
ODD1 5D         ld  e,l
ODD2 13         inc  de
ODD3 EDB0       ldir
ODD5 D1         pop  de
ODD6 C1         pop  bc
ODD7 E1         pop  hl
ODD8 CD130C     call 0C13          SCR NEXT LINE
ODDB 1D         dec  e
ODDC 20D9       jr   nz,0DB7      SCR FLOOD BOX
ODDE C9         ret

```

***** SCR CHAR INVERT

```

ODDF 78         ld  a,b
ODE0 A9         xor  c
ODE1 4F         ld  c,a
ODE2 CD640B     call 0B64          SCR CHAR POSITION
ODE5 1608       ld  d,08
ODE7 E5         push hl
ODE8 C5         push bc
ODE9 7E         ld  a,(hl)
ODEA A9         xor  c
ODEB 77         ld  (hl),a
ODEC CDF90B     call 0BF9          SCR NEXT BYTE
ODEF 10F8       djnz 0DE9
ODF1 C1         pop  bc

```

***** Adresser mémoire couleurs

```

ODF2 E1         pop  hl

```

SCREEN PACK

```

ODF3 CD130C      call OC13      SCR NEXT LINE
ODF6 15          dec d
ODF7 20EE        Jr nz,ODE7
ODF9 C9          ret

```

***** SCR HW ROLL

```

ODFA 4F          ld c,a
ODFB C5          push bc
ODFC 11D0FF      ld de,FFD0
ODFF 0630        ld b,30
OE01 CD240E      call OE24
OE04 C1          pop b_c
OE05 CDBA07      call 07BA      MC WAIT FLYBACK
OE08 78          ld a,b
OE09 B7          or a
OE0A 200D        Jr nz,OE19
OE0C 11B0FF      ld de,FFB0
OE0F CD370E      call OE37
OE12 110000      ld de,0000
OE15 0620        ld b,20
OE17 180B        Jr OE24
OE19 115000      ld de,0050
OE1C CD370E      call OE37
OE1F 11B0FF      ld de,FFB0
OE22 0620        ld b,20
OE24 2AC9B1      ld hl,(B1C9)
OE27 19          add hl,de
OE28 7C          ld a,h
OE29 E607        and 07
OE2B 67          ld h,a
OE2C 3ACBB1      ld a,(B1CB)   (High Byte Screen Start)
OE2F 84          add a,h
OE30 67          ld h,a
OE31 50          ld d,b
OE32 1E08        ld e,08
OE34 C3B70D      jp ODB7      SCR FLOOD BOX

OE37 2AC9B1      ld hl,(B1C9)
OE3A 19          add hl,de
OE3B C33C0B      jp OB3C     SCR SET OFFSET

```


SCREEN PACK

***** SCR SW ROLL

0E3E	F5	push	af	
0E3F	78	ld	a,b	
0E40	B7	or	a	
0E41	2830	jr	z,0E73	
0E43	E5	push	hl	
0E44	CD950B	call	0B95	
0E47	E3	ex	(sp),hl	
0E48	2C	inc	l	
0E49	CD640B	call	0B64	SCR CHAR POSITION
0E4C	4A	ld	c,d	
0E4D	7B	ld	a,e	
0E4E	D608	sub	08	
0E50	47	ld	b,a	
0E51	2817	jr	z,0E6A	
0E53	D1	pop	de	
0E54	CDBA07	call	07BA	MC WAIT FLYBACK
0E57	C5	push	bc	
0E58	E5	push	hl	
0E59	D5	push	de	
0E5A	CDA40E	call	0EA4	
0E5D	E1	pop	hl	
0E5E	CD130C	call	0C13	SCR NEXT LINE
0E61	EB	ex	de,hl	
0E62	E1	pop	hl	
0E63	CD130C	call	0C13	SCR NEXT LINE
0E66	C1	pop	bc	
0E67	10EE	djnz	0E57	
0E69	D5	push	de	
0E6A	E1	pop	hl	
0E6B	51	ld	d,c	
0E6C	1E08	ld	e,08	
0E6E	F1	pop	af	
0E6F	4F	ld	c,a	
0E70	C3B70D	jp	0DB7	SCR FLOOD BOX
0E73	E5	push	hl	
0E74	D5	push	de	
0E75	CD950B	call	0B95	
0E78	4A	ld	c,d	

SCREEN PACK

0E79	7B	ld	a,e	
0E7A	D608	sub	08	
0E7C	47	ld	b,a	
0E7D	D1	pop	de	
0E7E	E3	ex	(sp),h1	
0E7F	28E9	jr	z,0E6A	
0E81	C5	push	bc	
0E82	6B	ld	l,e	
0E83	54	ld	d,h	
0E84	1C	inc	e	
0E85	CD640B	call	0B64	SCR CHAR POSITION
0E88	EB	ex	de,h1	
0E89	CD640B	call	0B64	SCR CHAR POSITION
0E8C	C1	pop	bc	
0E8D	CDBA07	call	07BA	MC WAIT FLYBACK
0E90	CD2DOC	call	0C2D	SCR PREV LINE
0E93	E5	push	h1	
0E94	EB	ex	de,h1	
0E95	CD2DOC	call	0C2D	SCR PREV LINE
0E98	E5	push	h1	
0E99	C5	push	bc	
0E9A	CDA40E	call	0EA4	
0E9D	C1	pop	bc	
0E9E	D1	pop	de	
0E9F	E1	pop	h1	
0EA0	10EE	djnz	0E90	
0EA2	18C6	jr	0E6A	
0EA4	0600	ld	b,00	
0EA6	CDE60E	call	0EE6	
0EA9	3816	jr	c,0EC1	
0EAB	CDE60E	call	0EE6	
0EAE	3025	jr	nc,0ED5	
0EB0	C5	push	bc	
0EB1	AF	xor	a	
0EB2	95	sub	l	
0EB3	4F	ld	c,a	
0EB4	EDB0	ldir		
0EB6	C1	pop	bc	
0EB7	2F	cpl		
0EB8	3C	inc	a	

SCREEN PACK

OEB9	81	add	a,c	
OEBA	4F	ld	c,a	
OEBB	7C	ld	a,h	
OEBC	D608	sub	08	
OEBE	67	ld	h,a	
OEBF	1814	jr	OED5	
OEC1	CDE60E	call	OEE6	
OEC4	3812	jr	c,OED8	
OEC6	C5	push	bc	
OEC7	AF	xor	a	
OEC8	93	sub	e	
OEC9	4F	ld	c,a	
OECA	EDB0	ldir		
OECB	C1	pop	bc	
OECD	2F	cpl		
OECE	3C	inc	a	
OECF	81	add	a,c	
OED0	4F	ld	c,a	
OED1	7A	ld	a,d	
OED2	D608	sub	08	
OED4	57	ld	d,a	
OED5	EDB0	ldir		
OED7	C9	ret		
OED8	41	ld	b,c	
OED9	7E	ld	a,(hl)	
OEDA	12	ld	(de),a	
OEDB	CDF90B	call	OBF9	SCR NEXT BYTE
OEDE	EB	ex	de,hl	
OEDF	CDF90B	call	OBF9	SCR NEXT BYTE
OEE2	EB	ex	de,hl	
OEE3	10F4	djnz	OED9	
OEE5	C9	ret		
ç				
OEE6	79	ld	a,c	
OEE7	EB	ex	de,hl	
OEE8	3D	dec	a	
OEE9	85	add	a,l	
OEEA	D0	ret	nc	
OEEB	7C	ld	a,h	

SCREEN PACK

```
OEEC E607      and  07
OEEE EE07      xor  07
OEF0 C0        ret  nz
OEF1 37        scf
OEF2 C9        ret
```

***** SCR UNPACK

```
OEF3 CDECOA    call OAEC      SCR GET MODE
OEF6 0608      ld  b,08
OEF8 3831      jr  c,0F2B
OEF9 2806      jr  z,0F02
OEF0 010800    ld  bc,0008
OEF1 EDB0      ld  dir
OEF2 C9        ret

OF02 4E        ld  c,(h1)
OF03 23        inc  h1
OF04 E5        push h1
OF05 C5        push bc
OF06 0604      ld  b,04
OF08 21CFB1    ld  hl,B1CF    Cartes bits suivant Mode
OF09 AF        xor  a
OF0A CB01      rlc  c
OF0B 3001      jr  nc,0F11
OF0C B6        or   (h1)
OF0D 23        inc  h1
OF0E 10F8      djnz 0F0C
OF0F 12        ld  (de),a
OF10 13        inc  de
OF11 0604      ld  b,04
OF12 21CFB1    ld  hl,B1CF    Cartes bits suivant Mode
OF13 AF        xor  a
OF14 CB01      rlc  c

          3001      jr  nc,0F21
OF20 B6        or   (h1)
OF21 23        inc  h1
OF22 10F8      djnz 0F1C
OF23 12        ld  (de),a
OF24 13        inc  de
OF25 C1        pop  bc
OF26 C1
```

SCREEN PACK

```

OF27 E1      pop  hl
OF28 10D8    djnz 0F02
OF2A C9      ret

OF2B 4E      ld   c,(hl)
OF2C 23      inc  hl
OF2D E5      push hl
OF2E C5      push bc
OF2F 0604    ld   b,04
OF31 AF      xor  a
OF32 21CFB1  ld   hl,B1CF      Cartes bits suivant Mode
OF35 CB01    rlc  c
OF37 3001    jr   nc,0F3A
OF39 7E      ld   a,(hl)
OF3A 23      inc  hl
OF3B CB01    rlc  c
OF3D 3001    jr   nc,0F40
OF3F B6      or   (hl)
OF40 12      ld   (de),a
OF41 13      inc  de
OF42 10ED    djnz 0F31
OF44 C1      pop  bc
OF45 E1      pop  hl
OF46 10E3    djnz 0F2B
OF48 C9      ret

```

***** SCR REPACK

```

OF49 4F      ld   c,a
OF4A CD640B  call 0B64      SCR CHAR POSITION
OF4D CDEC0A  call 0AEC      SCR GET MODE
OF50 0608    ld   b,08
OF52 3845    jr   c,0F99
OF54 280B    jr   z,0F61
OF56 7E      ld   a,(hl)
OF57 A9      xor  c
OF58 2F      cpl
OF59 12      ld   (de),a
OF5A 13      inc  de
OF5B CD130C  call 0C13      SCR NEXT LINE
OF5E 10F6    djnz 0F56

```

SCREEN PACK

0F60	C9	ret	
0F61	E5	push	hl
0F62	D5	push	de
0F63	E5	push	hl
0F64	7E	ld	a,(hl)
0F65	A9	xor	c
0F66	21CFB1	ld	hl,B1CF
			Cartes bits suivant Mode
0F69	1604	ld	d,04
0F6B	F5	push	af
0F6C	A6	and	(hl)
0F6D	2001	jr	nz,0F70
0F6F	37	scf	
0F70	CB13	rl	e
0F72	23	inc	hl
0F73	F1	pop	af
0F74	15	dec	d
0F75	20F4	jr	nz,0F6B
0F77	E1	pop	hl
0F78	CD90B	call	0BF9
			SCR NEXT BYTE
0F7B	7E	ld	a,(hl)
0F7C	A9	xor	c
0F7D	21CFB1	ld	hl,B1CF
			Cartes bits suivant Mode
0F80	1604	ld	d,04
0F82	F5	push	af
0F83	A6	and	(hl)
0F84	2001	jr	nz,0F87
0F86	37	scf	
0F87	CB13	rl	e
0F89	23	inc	hl
0F8A	F1	pop	af
0F8B	15	dec	d
0F8C	20F4	jr	nz,0F82
0F8E	E1	pop	hl
0F8F	73	ld	(hl),e
0F90	EB	ex	de,hl
0F91	13	inc	de
0F92	E1	pop	hl
0F93	CD130C	call	0C13
			SCR NEXT LINE
0F96	10C9	djnz	0F61

SCREEN PACK

```

0F98 C9          ret
0F99 E5          push hl
0F9A D5          push de
0F9B 1604        ld d,04
0F9D 7E          ld a,(hl)
0F9E E5          push hl
0F9F A9          xor c
0FA0 F5          push af
0FA1 21CFB1      ld hl,B1CF      Cartes bits suivant Mode
0FA4 A6          and (hl)
0FA5 2001        jr nz,0FA8
0FA7 37          scf
0FA8 CB13        rl e
0FAA F1          pop af
0FAB 23          inc hl
0FAC A6          and (hl)
0FAD 2001        jr nz,0FB0
0FAF 37          scf
0FB0 CB13        rl e
0FB2 E1          pop hl
0FB3 CDF90B      call 0BF9      SCR NEXT BYTE
0FB6 15          dec d
0FB7 20E4        jr nz,0F9D
0FB9 E1          pop hl
0FBA 73          ld (hl),e
0FBB EB          ex de,hl
0FBC 13          inc de
0FBD E1          pop hl
0FBE CD130C      call 0C13      SCR NEXT LINE
0FC1 10D6        djnz 0F99
0FC3 C9          ret

```

***** SCR HORIZONTAL

```

0FC4 F5          push af
0FC5 E5          push hl
0FC6 7A          ld a,d
0FC7 2F          cpl
0FC8 67          ld h,a
0FC9 7B          ld a,e

```

SCREEN PACK

0FCA	2F	cpl	
0FCB	6F	ld	l,a
0FCC	23	inc	hl
0FCD	09	add	hl,bc
0FCE	23	inc	hl
0FCF	E3	ex	(sp),hl
0FDD	AF	xor	a
0FD1	93	sub	e
0FD2	F5	push	af
0FD3	CDA90B	call	OBA9 SCR DOT POSITION
0FD6	E5	push	hl
0FD7	78	ld	a,b
0FD8	2F	cpl	
0FD9	6F	ld	l,a
0FDA	26FF	ld	h,FF
0FDC	2207B2	ld	(B207),hl
0FDF	E1	pop	hl
0FE0	F1	pop	af
0FE1	A0	and	b
0FE2	47	ld	b,a
0FE3	2845	jr	z,102A
0FE5	E3	ex	(sp),hl
0FE6	1803	jr	0FEB
0FE8	1A	ld	a,(de)
0FE9	B1	or	c
0FEA	4F	ld	c,a
0FEB	2B	dec	hl
0FEC	7C	ld	a,h
0FED	B5	or	l
0FEE	2834	jr	z,1024
0FF0	13	inc	de
0FF1	10F5	djnz	0FE8
0FF3	EB	ex	de,hl
0FF4	E1	pop	hl
0FF5	F1	pop	af
0FF6	47	ld	b,a
0FF7	CDE8BD	call	BDE8 SCR WRITE
0FFA	CDF90B	call	0BF9 SCR NEXT BYTE
0FFD	E5	push	hl
0FFE	2A07B2	ld	hl,(B207)

SCREEN PACK

```

1001 19      add  hl,de
1002 300C    jr   nc,1010
1004 EB      ex   de,hl
1005 E1      pop  hl
1006 0EFF    ld   c,FF
1008 CDE8BD  call BDE8      SCR WRITE
100B CDF90B  call OBF9      SCR NEXT BYTE
100E 18ED    jr   OFFD
1010 7B      ld   a,e
1011 B7      or   a
1012 280E    jr   z,1022
1014 AF      xor  a
1015 21CFB1  ld   hl,B1CF    Cartes bits suivant Mode
1018 B6      or   (hl)
1019 23      inc  hl
101A 1D      dec  e
101B 20FB    jr   nz,1018
101D 4F      ld   c,a
101E E1      pop  hl
101F C3E8BD  jp   BDE8      SCR WRITE

1022 E1      pop  hl
1023 C9      ret

1024 E1      pop  hl
1025 F1      pop  af
1026 47      ld   b,a
1027 C3E8BD  jp   BDE8      SCR WRITE

102A D1      pop  de
102B F1      pop  af
102C 47      ld   b,a
102D 18CE    jr   OFFD

***** SCR VERTICAL
102F F5      push af
1030 E5      push hl
1031 7C      ld   a,h
1032 2F      cpl
1033 67      ld   h,a

```

SCREEN PACK

```

1034 7D      ld  a,l
1035 2F      cpl
1036 6F      ld  l,a
1037 23      inc hl
1038 09      add hl,bc
1039 23      inc hl
103A E3      ex  (sp),hl
103B CDA90B  call OBA9      SCR DOT POSITION
103E D1      pop de
103F F1      pop af
1040 47      ld  b,a
1041 CDE8BD  call BDE8      SCR WRITE
1044 CD2DOC  call OC2D      SCR PREV LINE
1047 1B      dec de
1048 7A      ld  a,d
1049 B3      or  e
104A 20F5    jr  nz,1041
104C C9      ret

```

***** Couleurs par défaut

```

104D 04 04 0A 13 0C 0B 14 15
1055 0D 06 1E 1F 07 12 19 04
105D 17 04 04 0A 13 0C 0B 14
1065 15 0D 06 1E 1F 07 12 19
106D 0A 07

```

```

106F C7      rst 0
1070 C7      rst 0
1071 C7      rst 0
1072 C7      rst 0
1073 C7      rst 0
1074 C7      rst 0
1075 C7      rst 0
1076 C7      rst 0
1077 C7      rst 0

```

2.5.5 TEXT SCREEN (TXT)

Ce pack est responsable de la gestion de textes, ce qui comprend également l'organisation des fenêtres.

Quelques remarques sont nécessaires en ce qui concerne la manipulation du curseur :

Les coordonnées réclamées ou fournies par les routines du curseur doivent être comprises comme des indications logiques, c'est-à-dire qu'elles se rapportent à la fenêtre actuelle. Les coordonnées 1,1 correspondent à l'angle supérieur gauche de la fenêtre. Si vous voulez par exemple positionner, avec TXT SET CURSOR, le curseur en dehors de la fenêtre, il sera automatiquement fixé sur la prochaine position possible à l'intérieur de la fenêtre, si le curseur est activé ou si un caractère doit être représenté ensuite.

La position actuelle (que vous pouvez lire avec TXT GET CURSOR) est ainsi également modifiée.

Si le curseur est désactivé, la nouvelle position souhaitée, jusqu'à ce qu'un caractère soit représenté ou jusqu'à ce que le curseur soit activé.

Deux routines permettent d'activer ou de désactiver le curseur. TXT CUR ON/OFF est une routine subordonnée à la routine TXT CUR ENABLE/DISABLE. Cela signifie que le curseur, après ENABLE, ne peut apparaître que s'il a été également autorisé avec ON.

Voici une routine que nous avons pas évoquée au chapitre 2.3:

TXT OUTPUT amène le caractère qui se trouve dans a sur le fenêtre actuelle de l'écran ou exécute ce caractère, s'il s'agit d'un caractère de commande.

Notez que cette routine utilise l'indirection TXT OUT ACTION! Si vous avez également détourné cette routine, c'est votre routine et non celle de la Rom qui sera utilisée.

TEXT SCREEN

*****TXT INITIALISE

```

1078 CD8810      call 1088      TXT RESET
107B AF          xor  a
107C 3295B2     ld  (B295),a
107F 210100     ld  h1,0001
1082 CD3D11     call 113D     TXT fixer param. défaut
1085 C3A310     jp  10A3     Reset Params (toutes les fenêtres)

```

*****TXT RESET

```

1088 219110     ld  h1,1091  Restore TXT Indirections
108B CD8A0A     call 0A8A    Move (h1+3)=>((h1+1)),cnt=(h1)
108E C35B14     jp  145B
1091 0F          db  0F       15 octets
1092 CDBD       dw  BDCD     adresse objet
1094 C36312     jp  1263     TXT DRAW/UNDRAW CURSOR
1097 C36312     jp  1263     TXT DRAW/UNDRAW CURSOR
109A C34A13     jp  134A     TXT WRITE CHAR
109D C3C013     jp  13C0     TXT UNWRITE
10A0 C30C14     jp  140C     TXT OUT ACTION

```

*****Reset Params (toutes fenêtres)

```

10A3 3E08       ld  a,08
10A5 110DB2     ld  de,B20D  Start Params Fenêtre 0
10A8 2185B2     ld  h1,B285  pos. curseur act.(Row,Col)
10AB 010F00     ld  bc,000F
10AE EDB0       ldir
10B0 3D          dec  a
10B1 20F5       jr  nz,10A8
10B3 320CB2     ld  (B20C),a (fenêtre écran act.)
10B6 C9          ret
10B7 3A0CB2     ld  a,(B20C) (fenêtre écran act.)
10BA 4F          ld  c,a
10BB 0608       ld  b,08
10BD 78          ld  a,b

```

TEXT SCREEN

```

10BE 3D          dec    a
10BF CDE810     call   10E8      TXT STR SELECT
10C2 CDD0BD     call   BDD0      TXT UNDRAW CURSOR
10C5 CDC312     call   12C3      TXT GET PAPER
10C8 3290B2     ld     (B290),a (TXT Paper act.)
10CB CDBD12     call   12BD      TXT GET PEN
10CE 328FB2     ld     (B28F),a (TXT Pen act.)
10D1 10EA      djnz   10BD
10D3 79         ld     a,c
10D4 C9         ret

10D5 4F         ld     c,a
10D6 0608      ld     b,08
10D8 78         ld     a,b
10D9 3D         dec    a
10DA CDE810     call   10E8      TXT STR SELECT
10DD C5         push  bc
10DE 2A8FB2     ld     hl,(B28F) (TXT Pen act.)
10E1 CD3D11     call   113D      TXT fixer paramètres défaut)
10E4 C1         pop   bc
10E5 10F1      djnz   10D8
10E7 79         ld     a,c

```

*****TXT STR SELECT

```

10E8 E607      and    07
10EA 210CB2     ld     hl,B20C  Fenêtre écran act.
10ED BE        cp     (hl)
10EE C8        ret    z
10EF C5        push  bc
10F0 D5        push  de
10F1 4E        ld     c,(hl)
10F2 77        ld     (hl),a
10F3 47        ld     b,a
10F4 79        ld     a,c
10F5 CD2A11     call   112A      Adr. params fenêtre => de
10F8 CD2211     call   1122      ldir cnt=15
10FB 78        ld     a,b
10FC CD2A11     call   112A      Adr. params fenêtre => de
10FF EB        ex     de,hl
1100 CD2211     call   1122      ldir cnt=15

```

TEXT SCREEN

```

1103 79      ld   a,c
1104 D1      pop  de
1105 C1      pop  bc
1106 C9      ret

```

*****TXT SWAP STREAMS

```

1107 3A0CB2  ld   a,(B20C) (fenêtre écran act.)
110A F5      push af
110B 79      ld   a,c
110C CDE810  call 10E8      TXT STR SELECT
110F 78      ld   a,b
1110 320CB2  ld   (B20C),a (fenêtre écran act.)
1113 CD2A11  call 112A      Adr. params fenêtre => de
1116 D5      push de
1117 79      ld   a,c
1118 CD2A11  call 112A      Adr. params fenêtre => de
111B E1      pop  hl
111C CD2211  call 1122      ldir cnt=15
111F F1      pop  af
1120 18C6    jr   10E8      TXT STR SELECT

```

*****ldir cnt=15

```

1122 C5      push bc
1123 010F00  ld   bc,000F
1126 EDB0    ldir
1128 C1      pop  bc
1129 C9      ret

```

***** Adr. params fenêtre => de

```

112A E607    and  07
112C 5F      ld   e,a
112D 87      add  a,a
112E 87      add  a,a
112F 87      add  a,a
1130 87      add  a,a
1131 93      sub  e
1132 C60D    add  a,0D
1134 5F      ld   e,a
1135 CEB2    adc  a,B2
1137 93      sub  e
1138 57      ld   d,a

```

TEXT SCREEN

1139 2185B2 ld hl,B285 pos. curseur act.(Row,Col)
 113C C9 ret

*****TXT fixer params défaut

113D EB ex de,hl
 113E 3E03 ld a,03
 1140 328DB2 ld (B28D),a (flag curseur act.)
 1143 7A ld a,d
 1144 CDAE12 call 12AE TXT SET PAPER
 1147 7B ld a,e
 1148 CDA912 call 12A9 TXT SET PEN
 114B AF xor a
 114C CDA713 call 13A7 TXT SET GRAPHIC
 114F CD7A13 call 137A TXT SET BACK
 1152 210000 ld hl,0000
 1155 117F7F ld de,7F7F
 1158 CDOC12 call 120C TXT WIN ENABLE
 115B C35114 jp 1451 TXT VDU ENABLE

*****TXT SET COLUMN

115E 3D dec a
 115F 2189B2 ld hl,B289 fenêtre act. gauche
 1162 86 add a,(hl)
 1163 2A85B2 ld hl,(B285) (Pos. curseur act.(Row,Col))
 1166 67 ld h,a
 1167 180E jr 1177

*****TXT SET ROW

1169 3D dec a
 116A 2188B2 ld hl,B288 fenêtre act. haut
 116D 86 add a,(hl)
 116E 2A85B2 ld hl,(B285) (Pos. curseur act.(Row,Col))
 1171 6F ld l,a
 1172 1803 jr 1177

*****TXT SET CURSOR

1174 CD8A11 call 118A lfd Fenst. haut,gauche+hl
 1177 CDD0BD call BDD0 TXT UNDRAW CURSOR
 117A 2285B2 ld (B285),hl (Pos. curseur act.(Row,Col))
 117D C3CDBD jp BDCD TXT DRAW CURSOR

TEXT SCREEN

```
***** TXT GET CURSOR
1180 2A85B2      ld   hl,(B285) (Pos. curseur act.(Row,Col))
1183 CD9711      call 1197      fenêtre act. haut,gauche-hl
1186 3A8CB2      ld   a,(B28C) (Roll Count act.)
1189 C9          ret
```

```
***** fenêtre act. haut,gauche+hl
118A 3A88B2      ld   a,(B288) (fenêtre act. haut)
118D 3D          dec   a
118E 85          add  a,l
118F 6F          ld   l,a
1190 3A89B2      ld   a,(B289) (fenêtre act. gauche)
1193 3D          dec   a
1194 84          add  a,h
1195 67          ld   h,a
1196 C9          ret
```

```
***** fenêtre act. haut,gauche-hl
1197 3A88B2      ld   a,(B288) (fenêtre act. haut)
119A 95          sub  l
119B 2F          cpl
119C 3C          inc  a
119D 3C          inc  a
119E 6F          ld   l,a
119F 3A89B2      ld   a,(B289) (fenêtre act. gauche)
11A2 94          sub  h
11A3 2F          cpl
11A4 3C          inc  a
11A5 3C          inc  a
11A6 67          ld   h,a
11A7 C9          ret
```

```
***** move Cursor
11A8 CDD0BD      call BDD0      TXT UNDRAW CURSOR
11AB 2A85B2      ld   hl,(B285) (Pos. curseur act.(Row,Col))
11AE CDDA11      call 11DA      hl dans limites fenêtre?
11B1 2285B2      ld   (B285),hl (Pos. curseur act.(Row,Col))
11B4 D8          ret   c
11B5 E5          push hl
11B6 218CB2      ld   hl,B28C  Act. Roll Count
```


TEXT SCREEN

```

11B9 78      ld   a,b
11BA 87      add  a,a
11BB 3C      inc  a
11BC 86      add  a,(hl)
11BD 77      ld   (hl),a
11BE CD5612  call 1256   TXT GET WINDOW
11C1 3A90B2  ld   a,(B290) (TXT act. Paper)
11C4 F5      push af
11C5 DC3E0E  call c,0E3E SCR SW ROLL
11C8 F1      pop  af
11C9 D4FA0D  call nc,ODFA SCR HW ROLL
11CC E1      pop  hl
11CD C9      ret

```

***** TXT VALIDATE

```

11CE CD8A11  call 118A   fenêtre act. haut,gauche+hl
11D1 CDDA11  call 11DA   hl dans limites fenêtre?
11D4 F5      push af
11D5 CD9711  call 1197   fenêtre act. haut,gauche-hl
11D8 F1      pop  af
11D9 C9      ret

```

***** hl dans limites fenêtre?

```

11DA 3A8BB2  ld   a,(B28B) (fenêtre act. droite)
11DD BC      cp   h
11DE F2E611  jp   p,11E6
11E1 3A89B2  ld   a,(B289) (fenêtre act. gauche)
11E4 67      ld   h,a
11E5 2C      inc  l
11E6 3A89B2  ld   a,(B289) (fenêtre act. gauche)
11E9 3D      dec  a
11EA BC      cp   h
11EB FAF311  jp   m,11F3
11EE 3A8BB2  ld   a,(B28B) (fenêtre act. droite)
11F1 67      ld   h,a
11F2 2D      dec  l
11F3 3A88B2  ld   a,(B288) (fenêtre act. haut)
11F6 3D      dec  a
11F7 BD      cp   l
11F8 F20612  jp   p,1206

```

TEXT SCREEN

```

11FB 3A8AB2      ld  a,(B28A) (fenêtre act. bas)
11FE BD         cp  l
11FF 37         scf
1200 F0         ret  p
1201 6F         ld  l,a
1202 06FF       ld  b,FF
1204 B7         or  a
1205 C9         ret

1206 3C         inc a
1207 6F         ld  l,a
1208 0600       ld  b,00
120A B7         or  a
120B C9         ret

```

***** TXT WIN ENABLE

```

120C CD570B     call 0B57      SCR CHAR LIMITS
120F 7C         ld  a,h
1210 CD4412     call 1244
1213 67         ld  h,a
1214 7A         ld  a,d
1215 CD4412     call 1244
1218 57         ld  d,a
1219 BC         cp  h
121A 3002       Jr  nc,121E
121C 54         ld  d,h
121D 67         ld  h,a
121E 7D         ld  a,l
121F CD4D12     call 124D
1222 6F         ld  l,a
1223 7B         ld  a,e
1224 CD4D12     call 124D
1227 5F         ld  e,a
1228 BD         cp  l
1229 3002       Jr  nc,122D
122B 5D         ld  e,l
122C 6F         ld  l,a
122D 2288B2     ld  (B288),h1 (fenêtre act. haut)
1230 ED538AB2  ld  (B28A),de (fenêtre act. bas)
1234 7C         ld  a,h

```

TEXT SCREEN

```

1235 B5      or    l
1236 2006    jr    nz,123E
1238 7A      ld    a,d
1239 A8      xor    b
123A 2002    jr    nz,123E
123C 7B      ld    a,e
123D A9      xor    c
123E 3287B2   ld    (B287),a (Flag fenêtre (0=écran entier))
1241 C37711   jp    1177

```

```

1244 B7      or    a
1245 F24912   jp    p,1249
1248 AF      xor    a
1249 B8      cp    b
124A D8      ret   c
124B 78      ld    a,b
124C C9      ret

```

```

124D B7      or    a
124E F25212   jp    p,1252
1251 AF      xor    a
1252 B9      cp    c
1253 D8      ret   c
1254 79      ld    a,c
1255 C9      ret

```

***** TXT GET WINDOW

```

1256 2A88B2   ld    hl,(B288) (fenêtre act. haut)
1259 ED5B8AB2 ld    de,(B28A) (fenêtre act. bas)
125D 3A87B2   ld    a,(B287) (flag fenêtre (0=écran entier))
1260 C6FF     add   a,FF
1262 C9      ret

```

***** TXT DRAW/UNDRAW CURSOR

```

1263 3A8DB2   ld    a,(B28D) (act. Cursor Flag)
1266 B7      or    a
1267 C0      ret   nz

```

***** TXT PLACE/REMOVE CURSOR

```

1268 C5      push bc

```

TEXT SCREEN

```

1269 D5      push  de
126A E5      push  hl
126B CDAB11  call  11AB
126E ED4B8FB2 ld   bc,(B28F) (TXT act. Pen)
1272 CDDF0D  call  ODDF      SCR CHAR INVERT
1275 E1      pop   hl
1276 D1      pop   de
1277 C1      pop   bc
1278 C9      ret

```

***** TXT CUR ON

```

1279 F5      push  af
127A 3EFD    ld   a,FD
127C CD8B12  call  128B      Cur Enable Cont'd
127F F1      pop   af
1280 C9      ret

```

***** TXT CUR OFF

```

1281 F5      push  af
1282 3E02    ld   a,02
1284 CD9C12  call  129C      Cur Disable Cont'd
1287 F1      pop   af
1288 C9      ret

```

***** TXT CUR ENABLE

```

1289 3EFE    ld   a,FE

```

***** Cur Enable Cont'd

```

128B F5      push  af
128C CDD0BD  call  BDD0      TXT UNDRAW CURSOR
128F F1      pop   af
1290 E5      push  hl
1291 218DB2  ld   hl,B28D   act. Cursor Flag
1294 A6      and   (hl)
1295 77      ld   (hl),a
1296 E1      pop   hl
1297 C3CDBD  jp   BDCD      TXT DRAW CURSOR

```

***** TXT CUR DISABLE

```

129A 3E01    ld   a,01

```

TEXT SCREEN

***** Cur Disable Cont'd0

```

129C F5      push af
129D CDD0BD  call BDD0    TXT UNDRAW CURSOR
12A0 F1      pop  af
12A1 E5      push hl
12A2 218DB2  ld   hl,B28D act. Cursor Flag
12A5 B6      or   (hl)
12A6 77      ld   (hl),a
12A7 E1      pop  hl
12A8 C9      ret

```

***** TXT SET PEN

```

12A9 218FB2  ld   hl,B28F  TXT act. Pen
12AC 1803   jr   12B1

```

***** TXT SET PAPER

```

12AE 2190B2  ld   hl,B290  TXT act. Paper
12B1 F5      push af
12B2 CDD0BD  call BDD0    TXT UNDRAW CURSOR
12B5 F1      pop  af
12B6 CD860C  call OC86    SCR INK ENCODE
12B9 77      ld   (hl),a
12BA C3CDBD  jp   BDCD    TXT DRAW CURSOR

```

***** TXT GET PEN

```

12BD 3A8FB2  ld   a,(B28F) (TXT act. Pen)
12C0 C3A00C  jp   OCA0    SCR INK DECODE

```

***** TXT GET PAPER

```

12C3 3A90B2  ld   a,(B290) (TXT act. Paper)
12C6 C3A00C  jp   OCA0    SCR INK DECODE

```

***** TXT INVERSE

```

12C9 2A8FB2  ld   hl,(B28F) (TXT act. Pen)
12CC 7C      ld   a,h
12CD 65      ld   h,l
12CE 6F      ld   l,a
12CF 228FB2  ld   (B28F),hl (TXT act. Pen)
12D2 C9      ret

```

TEXT SCREEN

***** TXT GET MATRIX

```

12D3 D5          push  de
12D4 5F          ld    e,a
12D5 CD2A13     call  132A      TXT GET M TABLE
12D8 3009       jr    nc,12E3
12DA 57          ld    d,a
12DB 7B          ld    a,e
12DC 92          sub   d
12DD 3F          ccf
12DE 3003       jr    nc,12E3
12E0 5F          ld    e,a
12E1 1803       jr    12E6
12E3 210038     ld    hl,3800
12E6 F5          push  af
12E7 1600       ld    d,00
12E9 EB          ex    de,hl
12EA 29          add   hl,hl
12EB 29          add   hl,hl
12EC 29          add   hl,hl
12ED 19          add   hl,de
12EE F1          pop   af
12EF D1          pop   de
12F0 C9          ret

```

***** TXT SET MATRIX

```

12F1 EB          ex    de,hl
12F2 CDD312     call  12D3      TXT GET MATRIX
12F5 D0          ret    nc
12F6 EB          ex    de,hl
12F7 010800     ld    bc,0008
12FA EDB0       ldir
12FC C9          ret

```

***** TXT SET M TABLE

```

12FD E5          push  hl
12FE 7A          ld    a,d
12FF B7          or    a
1300 1600       ld    d,00
1302 2019       jr    nz,131D
1304 15          dec   d

```

TEXT SCREEN

```

1305 D5      push  de
1306 4B      ld    c,e
1307 EB      ex    de,hl
1308 79      ld    a,c
1309 CDD312   call  12D3      TXT GET MATRIX
130C 7C      ld    a,h
130D AA      xor   d
130E 2004    jr    nz,1314
1310 7D      ld    a,l
1311 AB      xor   e
1312 2808    jr    z,131C
1314 C5      push  bc
1315 CDF712   call  12F7
1318 C1      pop   bc
1319 0C      inc   c
131A 20EC    jr    nz,1308
131C D1      pop   de
131D CD2A13   call  132A      TXT GET M TABLE
1320 ED5394B2 ld    (B294),de (1er Caractère User Matrix)
1324 D1      pop   de
1325 ED5396B2 ld    (B296),de (Adr. User Matrix)
1329 C9      ret

```

***** TXT GET M TABLE

```

132A 2A94B2   ld    hl,(B294) (1er Caractère User Matrix)
132D 7C      ld    a,h
132E 0F      rrca
132F 7D      ld    a,l
1330 2A96B2   ld    hl,(B296) (Adr. User Matrix)
1333 C9      ret

```

***** TXT WR CHAR

```

1334 47      ld    b,a
1335 3A8EB2   ld    a,(B28E) (VDU Flag (0=disabled))
1338 B7      or    a
1339 C8      ret   z
133A C5      push  bc
133B CDA811   call  11A8      move Cursor
133E 24      inc   h

```

TEXT SCREEN

```

133F 2285B2      ld  (B285),h1 (Pos. curseur act.(Row,Col))
1342 25          dec  h
1343 F1          pop  af
1344 CDD3BD      call BDD3      TXT WRITE CHAR
1347 C3CDBD      jp   BDCD      TXT DRAW CURSOR

```

***** TXT WRITE CHAR

```

134A E5          push h1
134B CDD312      call 12D3      TXT GET MATRIX
134E 1198B2      ld  de,B298
1351 D5          push de
1352 CDF30E      call 0EF3      SCR UNPACK
1355 D1          pop  de
1356 E1          pop  h1
1357 CD640B      call 0B64      SCR CHAR POSITION
135A 0E08        ld  c,08
135C C5          push bc
135D E5          push h1
135E C5          push bc
135F D5          push de
1360 EB          ex  de,h1
1361 4E          ld  c,(h1)
1362 CD7613      call 1376
1365 CDF90B      call 0BF9      SCR NEXT BYTE
1368 D1          pop  de
1369 13          inc  de
136A C1          pop  bc
136B 10F1        djnz 135E
136D E1          pop  h1
136E CD130C      call 0C13      SCR NEXT LINE
1371 C1          pop  bc
1372 0D          dec  c
1373 20E7        jr  nz,135C
1375 C9          ret

```

```

1376 2A91B2      ld  h1,(B291) (act. Background Mode)
1379 E9          jp  (h1)

```

***** TXT SET BACK

```

137A 219113      ld  h1,1391

```


TEXT SCREEN

```

137D B7      or   a
137E 2803    jr   z,1383
1380 219F13  ld   hl,139F
1383 2291B2  ld   (B291),hl (act. Background Mode)
1386 C9      ret

```

***** TXT GET BACK

```

1387 2A91B2  ld   hl,(B291) (act. Background Mode)
138A 116FEC  ld   de,EC6F
138D 19      add  hl,de
138E 7C      ld   a,h
138F B5      or   l
1390 C9      ret

```

```

1391 2A8FB2  ld   hl,(B28F) (TXT act. Pen)
1394 79      ld   a,c
1395 2F      cpl
1396 A4      and  h
1397 47      ld   b,a
1398 79      ld   a,c
1399 A5      and  l
139A B0      or   b
139B 0EFF    ld   c,FF
139D 1803    jr   13A2
139F 3A8FB2  ld   a,(B28F) (TXT act. Pen)
13A2 47      ld   b,a
13A3 EB      ex   de,hl
13A4 C36B0C jp   0C6B      SCR PIXELS

```

***** TXT SET GRAPHIC

```

13A7 3293B2  ld   (B293),a (GRA CHAR WR Mode (0=disabl))
13AA C9      ret

```

***** TXT RD CHAR

```

13AB E5      push hl
13AC D5      push de
13AD C5      push bc
13AE CDD0BD  call BDD0      TXT UNDRAW CURSOR
13B1 2A85B2  ld   hl,(B285) (Pos. curseur act.(Row,Col))
13B4 CDD6BD  call BDD6      TXT UNWRITE

```

TEXT SCREEN

```

13B7 F5          push af
13B8 CDCDBD     call BCD          TXT DRAW CURSOR
13BB F1          pop af
13BC C1          pop bc
13BD D1          pop de
13BE E1          pop hl
13BF C9          ret

```

***** TXT UNWRITE

```

13C0 3A8FB2     ld a,(B28F) (TXT act. Pen)
13C3 1198B2     ld de,B298
13C6 E5          push hl
13C7 D5          push de
13C8 CD490F     call OF49        SCR REPACK
13CB CDE313     call 13E3
13CE D1          pop de
13CF E1          pop hl
13D0 3001       jr nc,13D3
13D2 C0          ret nz
13D3 3A90B2     ld a,(B290) (TXT act. Paper)
13D6 D5          push de
13D7 CD490F     call OF49        SCR REPACK
13DA D1          pop de
13DB 0608       ld b,08
13DD 1A          ld a,(de)
13DE 2F          cpl
13DF 12          ld (de),a
13E0 13          inc de
13E1 10FA       djnz 13DD
13E3 0E00       ld c,00
13E5 79          ld a,c
13E6 CDD312     call 12D3        TXT GET MATRIX
13E9 1198B2     ld de,B298
13EC 0608       ld b,08
13EE 1A          ld a,(de)
13EF BE          cp (hl)
13F0 2009       jr nz,13FB
13F2 23          inc hl
13F3 13          inc de
13F4 10F8       djnz 13EE

```

TEXT SCREEN

```

13F6 79          ld  a,c
13F7 FE20        cp  20
13F9 37          scf
13FA C9          ret

13FB 0C          inc c
13FC 20E7        jr  nz,13E5
13FE AF          xor  a
13FF C9          ret

```

***** TXT OUTPUT

```

1400 F5          push af
1401 C5          pushf bc
1402 D5          push de
1403 E5          push hl
1404 CDD9BD       call BDD9      TXT OUT ACTION
1407 E1          pop  hl
1408 D1          pop  de
1409 C1          pop  bc
140A F1          pop  af
140B C9          ret

```

***** TXT OUT ACTION

```

140C 4F          ld  c,a
140D 3A93B2      ld  a,(B293)  (GRA CHAR WR Mode (0=disabl))
1410 B7          or  a
1411 79          ld  a,c
1412 C24519      jp  nz,1945   GRA WR CHAR
1415 21B8B2      ld  hl,B2B8   Compteur de caractères Control Buffer
1418 46          ld  b,(hl)
1419 78          ld  a,b
141A FE0A        cp  0A        Control Buffer plein ?
141C 3028        jr  nc,1446   oui =>
141E B7          or  a        vide ?
141F 2006        jr  nz,1427   non =>
1421 79          ld  a,c
1422 FE20        cp  20        caractère de commande?
1424 D23413      jp  nc,1334   non => TXT WR CHAR
1427 04          inc b        Compteur +1
1428 70          ld  (hl),b

```

TEXT SCREEN

```

1429 58          ld    e,b
142A 1600        ld    d,00
142C 19          add   hl,de
142D 71          ld    (hl),c
142E 3AB9B2      ld    a,(B2B9) (Start Control Buffer)
1431 5F          ld    e,a
1432 21C3B2      ld    hl,B2C3 Table de saut caractère de commande
1435 19          add   hl,de
1436 19          add   hl,de
1437 19          add   hl,de
1438 7E          ld    a,(hl) nombre requis
1439 B8          cp    b      atteint paramètre de commande ?
143A D0          ret   nc     non =>
143B 23          inc   hl
143C 5E          ld    e,(hl)
143D 23          inc   hl
143E 56          ld    d,(hl)
143F 21B9B2      ld    hl,B2B9 Start Control Buffer
1442 79          ld    a,c
1443 CD1600      call  0016 call (de)
1446 AF          xor   a
1447 32B8B2      ld    (B2B8),a (Compteur caractères Control Buffer)
144A C9          ret

```

***** TXT VDU DISABLE

```

144B CD9A12      call  129A TXT CUR DISABLE
144E AF          xor   a
144F 1805        jr    1456

```

***** TXT VDU ENABLE

```

1451 CD8912      call  1289 TXT CUR ENABLE
1454 3EFF        ld    a,FF
1456 328EB2      ld    (B28E),a (VDU Flag (0=disabled))
1459 18EB        jr    1446
145B AF          xor   a
145C 32B8B2      ld    (B2B8),a (compteur caractères Control Buffer)
145F 216B14      ld    hl,146B Saut défaut caractère de commande
1462 11C3B2      ld    de,B2C3 Table de saut caractère de commande
1465 016000      ld    bc,0060
1468 EDB0        ldir

```

TEXT SCREEN

146A C9 ret

***** Saut défaut Caractère de

commande

146B	00	db	00	
146C	E214	dw	14E2	00 aucun effet
146E	00	db	00	
146F	3413	dw	1334	01 TXT WR CHAR
1471	00	db	00	
1472	9A12	dw	129A	02 TXT CUR DISABLE
1474	00	db	00	
1475	8912	dw	1289	03 TXT CUR ENABLE
1477	01	db	01	
1478	CA0A	dw	0ACA	04 SCR SET MODE
147A	01	db	01	
147B	4519	dw	1945	05 GRA WR CHAR
147D	00	db	00	
147E	5114	dw	1451	06 TXT VDU ENABLE
1480	00	db	00	
1481	D814	dw	14D8	07 bip
1483	00	db	00	
1484	0A15	dw	150A	08 CRSR LEFT
1486	00	db	00	
1487	0F15	dw	150F	09 CRSR RGHT
1489	00	db	00	
148A	1415	dw	1514	0A CRSR DOWN
148C	00	db	00	
148D	1915	dw	1519	0B CRSR UP
148F	00	db	00	
1490	4015	dw	1540	0C TXT CLEAR WINDOW
1492	00	db	00	
1493	3015	dw	1530	0D CRSR sur début de ligne
1495	01	db	01	
1496	AE12	dw	12AE	0E TXT SET PAPER
1498	01	db	01	
1499	A912	dw	12A9	0F TXT SET PEN
149B	00	db	00	
149C	4F15	dw	154F	10 supprimer caractère sur CRS Pos
149E	00	db	00	
149F	8E15	dw	158E	11 supprimer ligne jusqu'à CRS Pos

TEXT SCREEN

14A1	00	db	00	
14A2	8415	dw	1584	12 supprimer ligne à partir de CRS
Pos				
14A4	00	db	00	
14A5	6D15	dw	156D	13 vider fenêtre Jusqu'à CRS Pos
14A7	00	db	00	
14A8	5615	dw	1556	14 vider fenêtre à partir de CRS Pos
14AA	00	db	00	
14AB	4B14	dw	144B	15 TXT VDU DISABLE
14AD	01	db	01	
14AE	E314	dw	14E3	16 Transparentmode mis/éteint
14B0	01	db	01	
14B1	490C	dw	0C49	17 SCR ACCESS
14B3	00	db	00	
14B4	C912	dw	12C9	18 TXT INVERSE
14B6	09	db	09	
14B7	0415	dw	1504	19 =SYMBOL (instruction)
14B9	04	db	04	
14BA	F814	dw	14F8	1A définir fenêtre
14BC	00	db	00	
14BD	E214	dw	14E2	1B aucun effet
14BF	03	db	03	
14C0	E814	dw	14E8	1C =INK (instruction)
14C2	02	db	02	
14C3	F114	dw	14F1	1D =BORDER (instruction)
14C5	00	db	00	
14C6	2A15	dw	152A	1E CRSR HOME
14C8	02	db	02	
14C9	3815	dw	1538	1F =LOCATE (instruction)

***** TXT GET CONTROLS

14CB	21C3B2	ld	hl,B2C3	Table de saut caractère de commande
14CE	C9	ret		
14CF	87	add	a,a	
14D0	00	nop		
14D1	00	nop		
14D2	5A	ld	e,d	
14D3	00	nop		
14D4	00	nop		

TEXT SCREEN

```
14D5 0B      dec  bc
14D6 14      inc  d
14D7 00      nop
```

***** 07 Bip

```
14D8 DDE5      push ix
14DA 21CF14    ld   hl,14CF
14DD CD9F1F    call 1F9F      SOUND QUEUE
14E0 DDE1      pop  ix
14E2 C9        ret
```

***** 16 Transparentmode mis/éteint

```
14E3 0F        rrca
14E4 9F        sbc  a,a
14E5 C37A13    jp   137A      TXT SET BACK
```

***** 1C =INK (instruction)

```
14E8 23        inc  hl
14E9 7E        ld   a,(hl)
14EA 23        inc  hl
14EB 46        ld   b,(hl)
14EC 23        inc  hl
14ED 4E        ld   c,(hl)
14EE C3EC0C    jp   0CEC      SCR SET INK
```

***** 1D =BORDER (instruction)

```
14F1 23        inc  hl
14F2 46        ld   b,(hl)
14F3 23        inc  hl
14F4 4E        ld   c,(hl)
14F5 C3F10C    jp   0CF1      SCR SET BORDER
```

***** 1A définir fenêtre

```
14F8 23        inc  hl
14F9 56        ld   d,(hl)
14FA 23        inc  hl
14FB 7E        ld   a,(hl)
14FC 23        inc  hl
14FD 5E        ld   e,(hl)
14FE 23        inc  hl
```

TEXT SCREEN

```

14FF 6E          ld  l,(h1)
1500 67          ld  h,a
1501 C30C12     jp   120C      TXT WIN ENABLE

```

***** 19 =SYMBOL (instruction)

```

1504 23          inc  h1
1505 7E          ld  a,(h1)
1506 23          inc  h1
1507 C3F112     jp   12F1      TXT SET MATRIX

```

***** 08 CRSR LEFT

```

150A 1100FF     ld  de,FF00
150D 180D       jr   151C

```

***** 09 CRSR RIGHT

```

150F 110001     ld  de,0100
1512 1808       jr   151C

```

***** 0A CRSR DOWN

```

1514 110100     ld  de,0001
1517 1803       jr   151C

```

***** 0B CRSR UP

```

1519 11FF00     ld  de,00FF
151C D5          push de
151D CDA811     call 11A8      move Cursor
1520 D1          pop  de
1521 7D          ld  a,l
1522 83          add  a,e
1523 6F          ld  l,a
1524 7C          ld  a,h
1525 82          add  a,d
1526 67          ld  h,a
1527 C37A11     jp   117A

```

***** 1E CRSR HOME

```

152A 2A88B2     ld  hl,(B288) (fenêtre act. haut)
152D C37711     jp   1177

```

***** 0D CRSR sur début de

TEXT SCREEN

ligne

```
1530 CDA811      call 11A8      move Cursor
1533 3A89B2      ld  a,(B289)  (fenêtre act. gauche)
1536 18EE        Jr   1526
```

***** 1F =LOCATE (instruction)

```
1538 23          inc  hl
1539 56          ld  d,(hl)
153A 23          inc  hl
153B 5E          ld  e,(hl)
153C EB          ex  de,hl
153D C37411      jp   1174      TXT SET CURSOR
```

***** TXT CLEAR WINDOW

```
1540 CDD0BD      call BDD0      TXT UNDRAW CURSOR
1543 2A88B2      ld  hl,(B288) (fenêtre act. haut)
1546 2285B2      ld  (B285),hl (Pos. curseur act.(Row,Col))
1549 ED5B8AB2    ld  de,(B28A) (fenêtre act. bas)
154D 1848        Jr   1597
```

***** 10 supprimer caractère sur CRS

Pos

```
154F CDA811      call 11A8      move Cursor
1552 54          ld  d,h
1553 5D          ld  e,l
1554 1841        Jr   1597
```

***** 14 vider fenêtre à partir de CRS Pos

```
1556 CD8415      call 1584      12 supprimer ligne à partir de CRS
Pos
1559 2A88B2      ld  hl,(B288) (fenêtre act. haut)
155C ED5B8AB2    ld  de,(B28A) (fenêtre act. bas)
1560 3A85B2      ld  a,(B285)  (Pos. curseur act.(Row,Col))
1563 6F          ld  l,a
1564 2C          inc  l
1565 BB          cp  e
1566 3A90B2      ld  a,(B290)  (TXT act. Paper)
1569 DCB30D      call c,0DB3    SCR FILL BOX
156C C9          ret
```

TEXT SCREEN

***** 13 vider fenêtre Jusqu'à CRS Pos

```

156D CD8E15      call 158E      11 supprimer ligne Jusqu'à CRS Pos
1570 2A88B2      ld   hl,(B288) (fenêtre act. haut)
1573 3A8BB2      ld   a,(B28B) (fenêtre act. droite)
1576 57          ld   d,a
1577 3A85B2      ld   a,(B285) (Pos. curseur act.(Row,Col))
157A 3D          dec  a
157B 5F          ld   e,a
157C BD          cp   l
157D 3A90B2      ld   a,(B290) (TXT act. Paper)
1580 D4B30D      call nc,0DB3  SCR FILL BOX
1583 C9          ret
    
```

***** 12 supprimer ligne à partir de CRS Pos

```

1584 CDA811      call 11A8      move Cursor
1587 5D          ld   e,l
1588 3A8BB2      ld   a,(B28B) (fenêtre act. droite)
158B 57          ld   d,a
158C 1809        jr   1597
    
```

***** 11 supprimer ligne jusqu'à CRS Pos

```

158E CDA811      call 11A8      move Cursor
1591 EB          ex   de,hl
1592 6B          ld   l,e
1593 3A89B2      ld   a,(B289) (fenêtre act. gauche)
1596 67          ld   h,a
1597 3A90B2      ld   a,(B290) (TXT act. Paper)
159A CDB30D      call 0DB3      SCR FILL BOX
159D CDCDBD      call BDCD      TXT DRAW CURSOR
15A0 C9          ret
    
```

```

15A1 C7          rst  0
15A2 C7          rst  0
15A3 C7          rst  0
15A4 C7          rst  0
15A5 C7          rst  0
15A6 C7          rst  0
15A7 C7          rst  0
15A8 C7          rst  0
15A9 C7          rst  0
    
```

TEXT SCREEN

15AA	C7	rst	0
15AB	C7	rst	0
15AC	C7	rst	0
15AD	C7	rst	0
15AE	C7	rst	0
15AF	C7	rst	0

GRAPHICS SCREEN

2.5.6 GRAPHICS SCREEN (GRA)

Ce pack sert exclusivement à la manipulation de la fenêtre graphique. Au sujet des indications de coordonnées qui sont réclamées par les différentes routines, il convient de faire les remarques suivantes:

Les coordonnées sont transmises en 3 (4) étapes. L'étape la plus proche de l'utilisateur est la position relativement à l'origine des coordonnées (ORIGIN) qu'il a lui-même fixée. Cette position est convertie en une position relativement à l'origine de l'écran (bas gauche).

Ces deux étapes dépendent du mode!

La dernière étape est l'adresse physique du point. Celle-ci dépend du mode actuel!

Une étape supplémentaire peut éventuellement être ajoutée auparavant, lorsqu'une paire de coordonnées relatives doit être convertie en une position absolue relativement à ORIGIN.

Les routines intéressantes sont: GRA PLOT ABSOLUTE qui fixe un point dans la position absolue fournie par de (coordonnée X) et hl (coordonnée Y), si ces coordonnées ne sortent pas de la fenêtre graphique.

Notez que cette routine fonctionne à travers l'indirection GRA PLOT au cours du déroulement de laquelle l'indirection SCR WRITE est également utilisée!

GRA LINE ABSOLUTE dessine une ligne à partir du curseur graphique actuel jusqu'à la position absolue déterminée par de (coordonnée X) et hl (coordonnée Y), si cette position ne sort pas du cadre de la fenêtre graphique. Ici aussi des indirections sont utilisées: GRA LINE et SCR WRITE!

GRA WR CHAR amène le caractère contenu dans a sur l'écran et ce dans la position actuelle du curseur GRAPHIQUE. Celle-ci détermine l'angle supérieur gauche du caractère. Le curseur graphique est ensuite déplacé de la distance correspondant à la largeur du caractère. Cette distance dépend du mode!

GRAPHICS SCREEN

***** GRA INITIALISE

```

15B0 CDDF15      call 15DF      GRA RESET
15B3 210100      ld  hl,0001    Pen 1 , Paper 0
15B6 7C          ld  a,h
15B7 CDFD17      call 17FD      GRA SET PAPER
15BA 7D          ld  a,l
15BB CDF617      call 17F6      GRA SET PEN
15BE 210000      ld  hl,0000    Origin sur 0,0
15C1 54          ld  d,h
15C2 5D          ld  e,l
15C3 CD0416      call 1604      GRA SET ORIGIN
15C6 110080      ld  de,8000
15C9 21FF7F      ld  hl,7FFF
15CC E5          push hl
15CD D5          push de
15CE CD3417      call 1734      GRA WIN WIDTH
15D1 E1          pop  hl
15D2 D1          pop  de
15D3 C37917      jp  1779      GRA WIN HEIGHT

15D6 C0A18       call 180A      GRA GET PAPER
15D9 67          ld  h,a
15DA CD0418      call 1804      GRA GET PEN
15DD 6F          ld  l,a
15DE C9          ret

```

***** GRA RESET

```

15DF 21E515      ld  hl,15E5    Restore GRA Indirections
15E2 C38A0A      jp  0A8A      Move (hl+3)=>((hl+1)),cnt=(hl)

15E5 09          db  09         9 octets
15E6 DCBD        dw  BDDC      Adresse objet
15E8 C31618      jp  1816      GRA PLOT

15EB C32A18      jp  182A      GRA TEST

15EE C33C18      jp  183C      GRA LINE

```

***** GRA MOVE RELATIVE

```

15F1 CD5716      call 1657      Add coord. act. + coord. rel.

```

GRAPHICS SCREEN

```
15F4 ED532CB3      ld      (B32C),de (Coord. X act.)
15F8 222EB3        ld      (B32E),hl (Coord. Y act.)
15FB C9            ret
```

***** GRA ASK CURSOR

```
15FC ED5B2CB3      ld      de,(B32C) (Coord. X act.)
1600 2A2EB3        ld      hl,(B32E) (Coord. Y act.)
1603 C9            ret
```

```
1604 ED5328B3      ld      (B328),de (X Origin)
1608 222AB3        ld      (B32A),hl (Y Origin)
160B 110000        ld      de,0000
160E 62            ld      h,d
160F 6B            ld      l,e
1610 18E2          jr      15F4      GRA MOVE ABSOLUTE
```

***** GRA GET ORIGIN

```
1612 ED5B28B3      ld      de,(B328) (X Origin)
1616 2A2AB3        ld      hl,(B32A) (Y Origin)
1619 C9            ret
```

***** aller chercher position de départ physique

```
161A CDFC15        call 15FC      GRA ASK CURSOR
```

***** aller chercher position objet physique et fixer Cur

```
161D CDF415        call 15F4      GRA MOVE ABSOLUTE
1620 E5            push hl
1621 CDECOA        call OAE C     SCR GET MODE
1624 2F            cpl
1625 C601          add a,01
1627 CE02          adc a,02
1629 2600          ld h,00
162B 6F            ld l,a
162C CB7A          bit 7,d
162E 2803          jr z,1633
1630 EB            ex de,hl
1631 19            add hl,de
1632 EB            ex de,hl
1633 2F            cpl
1634 A3            and e
```

GRAPHICS SCREEN

```

1635 5F      ld    e,a
1636 7D      ld    a,l
1637 2A28B3  ld    hl,(B328) (X Origin)
163A 19      add   hl,de
163B 0F      rrca
163C DC7417  call  c,1774
163F 0F      rrca
1640 DC7417  call  c,1774
1643 D1      pop   de
1644 E5      push  hl
1645 7A      ld    a,d
1646 07      rca
1647 3001     jr    nc,164A
1649 13      inc   de
164A 7B      ld    a,e
164B E6FE     and   FE
164D 5F      ld    e,a
164E 2A2AB3  ld    hl,(B32A) (Y Origin)
1651 19      add   hl,de
1652 CD7417  call  1774
1655 D1      pop   de
1656 C9      ret

```

***** Add coord. act. + coord. rel.

```

1657 E5      push  hl
1658 2A2CB3  ld    hl,(B32C) (coord. X act.)
165B 19      add   hl,de
165C D1      pop   de
165D E5      push  hl
165E 2A2EB3  ld    hl,(B32E) (coord. Y act.)
1661 19      add   hl,de
1662 D1      pop   de
1663 C9      ret

1664 D5      push  de
1665 E5      push  hl
1666 2A30B3  ld    hl,(B330) (Coord. X Fenêtre GRA Gauche)
1669 2B      dec   hl
166A B7      or    a

```

GRAPHICS SCREEN

166B	ED52	sbc	hl,de
166D	F2AC16	jp	p,16AC
1670	2A32B3	ld	hl,(B332) (Coord. X Fenêtre GRA droite)
1673	B7	or	a
1674	ED52	sbc	hl,de
1676	FAAC16	jp	m,16AC
1679	D1	pop	de
167A	2A34B3	ld	hl,(B334) (Coord. Y Fenêtre GRA Haut)
167D	B7	or	a
167E	ED52	sbc	hl,de
1680	FAAD16	jp	m,16AD
1683	2A36B3	ld	hl,(B336) (Coord. Y Fenêtre GRA Bas)
1686	2B	dec	hl
1687	B7	or	a
1688	ED52	sbc	hl,de
168A	FA9116	jp	m,1691
168D	ED5B36B3	ld	de,(B336) (Coord. Y Fenêtre GRA bas)
1691	2A36B3	ld	hl,(B336) (Coord. Y Fenêtre GRA bas)
1694	2B	dec	hl
1695	B7	or	a
1696	ED42	sbc	hl,bc
1698	F2AD16	jp	p,16AD
169B	2A34B3	ld	hl,(B334) (Coord. Y Fenêtre GRA haut)
169E	B7	or	a
169F	ED42	sbc	hl,bc
16A1	F2A816	jp	p,16A8
16A4	ED4B34B3	ld	bc,(B334) (Coord. Y Fenêtre GRA haut)
16A8	EB	ex	de,hl
16A9	D1	pop	de
16AA	37	scf	
16AB	C9	ret	
16AC	E1	pop	hl
16AD	D1	pop	de
16AE	B7	or	a
16AF	C9	ret	
16B0	E5	push	hl
16B1	D5	push	de
16B2	EB	ex	de,hl

GRAPHICS SCREEN

16B3	2A36B3	ld	hl,(B336) (Coord. Y Fenêtre GRA bas)
16B6	2B	dec	hl
16B7	B7	or	a
16B8	ED52	sbc	hl,de
16BA	F2F816	jp	p,16F8
16BD	2A34B3	ld	hl,(B334) (Coord. Y Fenêtre GRA haut)
16C0	B7	or	a
16C1	ED52	sbc	hl,de
16C3	FAF816	jp	m,16F8
16C6	D1	pop	de
16C7	2A32B3	ld	hl,(B332) (Coord. X Fenêtre GRA droite)
16CA	B7	or	a
16CB	ED52	sbc	hl,de
16CD	FAF916	jp	m,16F9
16D0	2A30B3	ld	hl,(B330) (Coord. X Fenêtre GRA gauche)
16D3	2B	dec	hl
16D4	B7	or	a
16D5	ED52	sbc	hl,de
16D7	FADE16	jp	m,16DE
16DA	ED5B30B3	ld	de,(B330) (Coord. X Fenêtre GRA gauche)
16DE	2A30B3	ld	hl,(B330) (Coord. X Fenêtre GRA gauche)
16E1	2B	dec	hl
16E2	B7	or	a
16E3	ED42	sbc	hl,bc
16E5	F2F916	jp	p,16F9
16E8	2A32B3	ld	hl,(B332) (Coord. X Fenêtre GRA droite)
16EB	B7	or	a
16EC	ED42	sbc	hl,bc
16EE	F2F516	jp	p,16F5
16F1	ED4B32B3	ld	bc,(B332) (Coord. X Fenêtre GRA droite)
16F5	E1	pop	hl
16F6	37	scf	
16F7	C9	ret	
16F8	D1	pop	de
16F9	E1	pop	hl
16FA	B7	or	a
16FB	C9	ret	
16FC	CD1D16	call	161D aller chercher pos obj phys

GRAPHICS SCREEN

```

16FF E5          push hl          et fixer Cur
1700 2A30B3      ld hl,(B330) (Coord. X Fenêtre GRA gauche)
1703 2B          dec hl
1704 B7          or a
1705 ED52        sbc hl,de
1707 F22D17      jp p,172D
170A 2A32B3      ld hl,(B332) (Coord. X Fenêtre GRA droite)
170D B7          or a
170E ED52        sbc hl,de
1710 FA2D17      jp m,172D
1713 E1          pop hl
1714 D5          push de
1715 EB          ex de,hl
1716 2A36B3      ld hl,(B336) (Coord. Y Fenêtre GRA bas)
1719 2B          dec hl
171A B7          or a
171B ED52        sbc hl,de
171D F23017      jp p,1730
1720 2A34B3      ld hl,(B334) (Coord. Y Fenêtre GRA haut)
1723 B7          or a
1724 ED52        sbc hl,de
1726 FA3017      jp m,1730
1729 EB          ex de,hl
172A D1          pop de
172B 37          scf
172C C9          ret

172D E1          pop hl
172E B7          or a
172F C9          ret

1730 EB          ex de,hl
1731 D1          pop de
1732 B7          or a
1733 C9          ret

```

***** GRA WIN WIDTH

```

1734 E5          push hl
1735 CD6017      call 1760
1738 D1          pop de

```

GRAPHICS SCREEN

```

1739 E5          push  hl
173A CD6017     call  1760
173D D1         pop   de
173E 7B        ld    a,e
173F 95        sub   l
1740 7A        ld    a,d
1741 9C        sbc   a,h
1742 3801      Jr    c,1745
1744 EB        ex    de,hl
1745 7B        ld    a,e
1746 E6F8      and   F8
1748 5F        ld    e,a
1749 7D        ld    a,l
174A F607      or    07
174C 6F        ld    l,a
174D CDECOA     call  OAEC          SCR GET MODE
1750 3D        dec   a
1751 FC7017     call  m,1770
1754 3D        dec   a
1755 FC7017     call  m,1770
1758 ED5330B3   ld    (B330),de (Coord. X Fenêtre GRA gauche)
175C 2232B3     ld    (B332),hl (Coord. X Fenêtre GRA droite)
175F C9        ret

1760 7A        ld    a,d
1761 B7        or    a
1762 210000     ld    hl,0000
1765 F8        ret   m
1766 217F02     ld    hl,027F
1769 7B        ld    a,e
176A 95        sub   l
176B 7A        ld    a,d
176C 9C        sbc   a,h
176D D0        ret   nc
176E EB        ex    de,hl
176F C9        ret

1770 CB2A      sra   d
1772 CB1B      rr    e
1774 CB2C      sra   h

```

GRAPHICS SCREEN

1776 CB1D rr l
 1778 C9 ret

***** GRA WIN HEIGHT

1779 E5 push hl
 177A CD9217 call 1792
 177D D1 pop de
 177E E5 push hl
 177F CD9217 call 1792
 1782 D1 pop de
 1783 7D ld a,l
 1784 93 sub e
 1785 7C ld a,h
 1786 9A sbc a,d
 1787 3801 Jr c,178A
 1789 EB ex de,hl
 178A ED5334B3 ld (B334),de (Coord. Y Fenêtre GRA haut)
 178E 2236B3 ld (B336),hl (Coord. Y Fenêtre GRA bas)
 1791 C9 ret

1792 7A ld a,d
 1793 B7 or a
 1794 210000 ld hl,0000
 1797 F8 ret m
 1798 CB3A srl d
 179A CB1B rr e
 179C 21C700 ld hl,00C7
 179F 7B ld a,e
 17A0 95 sub l
 17A1 7A ld a,d
 17A2 9C sbc a,h
 17A3 D0 ret nc
 17A4 EB ex de,hl
 17A5 C9 ret

***** GRA GET W WIDTH

17A6 ED5B30B3 ld de,(B330) (Coord. X Fenêtre GRA gauche)
 17AA 2A32B3 ld hl,(B332) (Coord. X Fenêtre GRA droite)
 17AD CDECOA call OAECS CR GET MODE

GRAPHICS SCREEN

```

17B0 3D          dec  a
17B1 FCB617     call m,17B6
17B4 3D          dec  a
17B5 F0          ret  p
17B6 29          add  hl,hl
17B7 23          inc  hl
17B8 EB          ex   de,hl
17B9 29          add  hl,hl
17BA EB          ex   de,hl
17BB C9          ret

```

***** GRA GET W HEIGHT

```

17BC ED5B34B3   ld   de,(B334) (Coord. Y Fenêtre GRA haut)
17CO 2A36B3     ld   hl,(B336) (Coord. Y Fenêtre GRA bas)
17C3 18F1       jr   17B6

```

***** GRA CLEAR WINDOW

```

17C5 CDA617     call 17A6      GRA GET W WIDTH
17C8 B7          or   a
17C9 ED52       sbc  hl,de
17CB 23          inc  hl
17CC CD7417     call 1774
17CF CD7417     call 1774
17D2 CB3D       srl  l
17D4 45         ld   b,l
17D5 ED5B36B3   ld   de,(B336) (Coord. Y Fenêtre GRA bas)
17D9 2A34B3     ld   hl,(B334) (Coord. Y Fenêtre GRA haut)
17DC E5         push hl
17DD B7          or   a
17DE ED52       sbc  hl,de
17E0 23          inc  hl
17E1 4D         ld   c,l
17E2 ED5B30B3   ld   de,(B330) (Coord. X Fenêtre GRA gauche)
17E6 E1         pop  hl
17E7 C5         push bc
17E8 CDA90B     call 0BA9     SCR DOT POSITION
17EB D1         pop  de
17EC 3A39B3     ld   a,(B339) (GRA Paper)
17EF 4F         ld   c,a
17F0 CDB70D     call 0DB7     SCR FLOOD BOX

```

GRAPHICS SCREEN

17F3 C30B16 jp 160B

***** GRA SET PEN

17F6 CD860C call OC86 SCR INK ENCODE
 17F9 3238B3 ld (B338),a (GRA Pen)
 17FC C9 ret

***** GRA SET PAPER

17FD CD860C call OC86 SCR INK ENCODE
 1800 3239B3 ld (B339),a (GRA Paper)
 1803 C9 ret

***** GRA GET PEN

1804 3A38B3 ld a,(B338) (GRA Pen)
 1807 C3A00C jp OCAO SCR INK DECODE

***** GRA GET PAPER

180A 3A39B3 ld a,(B339) (GRA Paper)
 180D C3A00C jp OCAO SCR INK DECODE

***** GRA PLOT RELATIVE

1810 CD5716 call 1657 Add coord. act. + coord. rel.

***** GRA PLOT ABSOLUTE

1813 C3DCBD jp BDDC GRA PLOT

***** GRA PLOT

1816 CDFC16 call 16FC
 1819 D0 ret nc
 181A CDA90B call 0BA9 SCR DOT POSITION
 181D 3A38B3 ld a,(B338) (GRA Pen)
 1820 47 ld b,a
 1821 C3E8BD jp BDE8 SCR WRITE

***** GRA TEST RELATIVE

1824 CD5716 call 1657 Add coord. act. + coord. rel.

***** GRA TEST ABSOLUTE

1827 C3DFBD jp BDDF GRA TEST

GRAPHICS SCREEN

***** GRA TEST

```

182A CDFC16      call 16FC
182D D20A18      jp  nc,180A    GRA GET PAPER
1830 CDA90B      call OBA9      SCR DOT POSITION
1833 C3E5BD      jp  BDE5      SCR READ
    
```

***** GRA LINE RELATIVE

```

1836 CD5716      call 1657      Add coord. act. + coord. rel.
    
```

***** GRA LINE ABSOLUTE

```

1839 C3E2BD      jp  BDE2      GRA LINE
    
```

***** GRA LINE

```

183C E5          push hl
183D D5          push de
183E CD1A16      call 161A      aller chercher pos départ phys
1841 ED5342B3   ld  (B342),de (Buffer de calcul Coord. X)
1845 2244B3     ld  (B344),hl (Buffer de calcul Coord. Y)
1848 D1         pop  de
1849 E1         pop  hl
184A CD1D16      call 161D      aller chercher pos obj phys
184D E5         push hl      et fixer Cur
184E 2A42B3     ld  hl,(B342) (Buffer de calcul Coord. X)
1851 B7         or   a
1852 ED52       sbc  hl,de
1854 44         ld  b,h
1855 4D         ld  c,l
1856 FA6918     jp  m,1869
1859 2A42B3     ld  hl,(B342) (Buffer de calcul Coord. X)
185C EB         ex  de,hl
185D 2242B3     ld  (B342),hl (Buffer de calcul Coord. X)
1860 2A44B3     ld  hl,(B344) (Buffer de calcul Coord. Y)
1863 E3         ex  (sp),hl
1864 2244B3     ld  (B344),hl (Buffer de calcul Coord. Y)
1867 1808       jr  1871
1869 210000     ld  hl,0000
186C B7         or   a
186D ED42       sbc  hl,bc
186F 44         ld  b,h
1870 4D         ld  c,l
    
```

GRAPHICS SCREEN

```

1871 D1      pop  de
1872 2A44B3  ld   h1,(B344) (Buffer de calcul Coord. Y)
1875 B7      or   a
1876 ED52   sbc  h1,de
1878 EB     ex   de,h1
1879 F28E18 jp   p,188E
187C 210000 ld   h1,0000
187F B7      or   a
1880 ED52   sbc  h1,de
1882 54     ld   d,h
1883 5D     ld   e,l
1884 B7     or   a
1885 ED42   sbc  h1,bc
1887 210100 ld   h1,0001
188A 3027   jr   nc,18B3
188C 180A   jr   1898
188E 62     ld   h,d
188F 6B     ld   l,e
1890 B7     or   a
1891 ED42   sbc  h1,bc
1893 21FFFF  ld   h1,FFFF
1896 3009   jr   nc,18A1
1898 223AB3  ld   (B33A),h1
189B 60     ld   h,b
189C 69     ld   l,c
189D 3EFF   ld   a,FF
189F 1819   jr   18BA
18A1 E5     push h1
18A2 2A42B3 ld   h1,(B342) (Buffer de calcul Coord. X)
18A5 09     add  h1,bc
18A6 2242B3 ld   (B342),h1 (Buffer de calcul Coord. X)
18A9 2A44B3 ld   h1,(B344) (Buffer de calcul Coord. Y)
18AC B7     or   a
18AD ED52   sbc  h1,de
18AF 2244B3 ld   (B344),h1 (Buffer de calcul Coord. Y)
18B2 E1     pop  h1
18B3 223AB3 ld   (B33A),h1
18B6 60     ld   h,b
18B7 69     ld   l,c
18B8 EB     ex   de,h1

```


GRAPHICS SCREEN

```

18B9 AF          xor  a
18BA 3246B3     ld   (B346),a
18BD 13         inc  de
18BE ED5340B3  ld   (B340),de
18C2 23         inc  hl
18C3 CD8C37     call 378C      hl/de => hl, Rest => de
18C6 223CB3     ld   (B33C),hl
18C9 ED533EB3  ld   (B33E),de
18CD ED4B40B3  ld   bc,(B340)
18D1 50         ld   d,b
18D2 59         ld   e,c
18D3 CB3A      srl  d
18D5 CB1B      rr   e
18D7 C5        push bc
18D8 ED4B3CB3  ld   bc,(B33C)
18DC 2A3EB3     ld   hl,(B33E)
18DF 19        add  hl,de
18E0 EB        ex  de,hl
18E1 2A40B3     ld   hl,(B340)
18E4 B7        or  a
18E5 ED52      sbc  hl,de
18E7 3007      jr   nc,18F0
18E9 19        add  hl,de
18EA EB        ex  de,hl
18EB B7        or  a
18EC ED52      sbc  hl,de
18EE EB        ex  de,hl
18EF 03        inc  bc
18F0 D5        push de
18F1 3A46B3     ld   a,(B346)
18F4 B7        or  a
18F5 2823      jr   z,191A
18F7 2A42B3     ld   hl,(B342) (Buffer de calcul Coord. X)
18FA 54        ld   d,h
18FB 5D        ld   e,l
18FC 09        add  hl,bc
18FD 2242B3     ld   (B342),hl (Buffer de calcul Coord. X)
1900 44        ld   b,h
1901 4D        ld   c,l
1902 0B        dec  bc

```

GRAPHICS SCREEN

```

1903 2A44B3      ld  hl,(B344) (Buffer de calcul Coord. Y)
1906 E5          push hl
1907 CDB016      call 16B0
190A 3A38B3      ld  a,(B338) (GRA Pen)
190D DCC40F      call c,0FC4 SCR HORIZONTAL
1910 D1          pop  de
1911 2A3AB3      ld  hl,(B33A)
1914 19          add  hl,de
1915 2244B3      ld  (B344),hl (Buffer de calcul Coord. Y)
1918 1823        jr   193D
191A 2A44B3      ld  hl,(B344) (Buffer de calcul Coord. Y)
191D 54          ld  d,h
191E 5D          ld  e,l
191F 09          add  hl,bc
1920 2244B3      ld  (B344),hl (Buffer de calcul Coord. Y)
1923 44          ld  b,h
1924 4D          ld  c,l
1925 0B          dec  bc
1926 EB          ex   de,hl
1927 ED5B42B3     ld  de,(B342) (Buffer de calcul Coord. X)
192B D5          push de
192C CD6416      call 1664
192F 3A38B3      ld  a,(B338) (GRA Pen)
1932 DC2F10      call c,102F SCR VERTICAL
1935 D1          pop  de
1936 2A3AB3      ld  hl,(B33A)
1939 19          add  hl,de
193A 2242B3      ld  (B342),hl (Buffer de calcul Coord. X)
193D D1          pop  de
193E C1          pop  bc
193F 0B          dec  bc
1940 78          ld  a,b
1941 B1          or   c
1942 2093        jr   nz,18D7
1944 C9          ret

```

***** GRA WR CHAR

```

1945 DDE5          push ix
1947 CDD312        call 12D3      TXT GET MATRIX
194A 113AB3        ld  de,B33A

```

GRAPHICS SCREEN

194D	D5	push	de	
194E	DDE1	pop	ix	
1950	010800	ld	bc,0008	
1953	EDB0	ldir		
1955	CD1A16	call	161A	aller chercher pos départ phys
1958	CDF16	call	16FF	
195B	304C	jr	nc,19A9	
195D	E5	push	hl	
195E	D5	push	de	
195F	010700	ld	bc,0007	
1962	EB	ex	de,hl	
1963	09	add	hl,bc	
1964	EB	ex	de,hl	
1965	B7	or	a	
1966	ED42	sbc	hl,bc	
1968	CDF16	call	16FF	
196B	D1	pop	de	
196C	E1	pop	hl	
196D	303A	jr	nc,19A9	
196F	CDA90B	call	0BA9	SCR DOT POSITION
1972	1608	ld	d,08	
1974	E5	push	hl	
1975	1E08	ld	e,08	
1977	CDCF19	call	19CF	
197A	CB09	rrc	c	
197C	DCF90B	call	c,0BF9	SCR NEXT BYTE
197F	DDCB0006	rlc	(ix+00)	
1986	E1	pop	hl	
1987	CD130C	call	0C13	SCR NEXT LINE
198A	DD23	inc	ix	
198C	15	dec	d	
198D	20E5	jr	nz,1974	
198F	DDE1	pop	ix	
1991	CD15FC	call	15FC	GRA ASK CURSOR
1994	EB	ex	de,hl	
1995	CDECOA	call	0AEC	SCR GET MODE
1998	010800	ld	bc,0008	
199B	FE01	cp	01	
199D	2804	jr	z,19A3	
199F	3003	jr	nc,19A4	

GRAPHICS SCREEN

19A1	09	add	hl, bc	
19A2	09	add	hl, bc	
19A3	09	add	hl, bc	
19A4	09	add	hl, bc	
19A5	EB	ex	de, hl	
19A6	C3F415	jp	15F4	GRA MOVE ABSOLUTE
19A9	0E08	ld	c, 08	
19AB	D5	push	de	
19AC	0608	ld	b, 08	
19AE	CDFF16	call	16FF	
19B1	300C	jr	nc, 19BF	
19B3	E5	push	hl	
19B4	D5	push	de	
19B5	C5	push	bc	
19B6	CDA90B	call	0BA9	SCR DOT POSITION
19B9	CDCF19	call	19CF	
19BC	C1	pop	bc	
19BD	D1	pop	de	
19BE	E1	pop	hl	
19BF	DDCB0006	rlc	(ix+00)	
19C6	D1	pop	de	
19C7	2B	dec	hl	
19C8	DD23	inc	ix	
19CA	0D	dec	c	
19CB	20DE	jr	nz, 19AB	
19CD	18C0	jr	198F	
19CF	DDCB007E	bit	7, (ix+00)	
19D8	3A39B3	ld	a, (B339)	(GRA Paper)
19DB	47	ld	b, a	
19DC	C3E8BD	jp	BDE8	SCR WRITE
19DF	C7	rst	0	

2.5.7 KEYBOARD MANAGER (KM)

Ce pack a pour fonction la surveillance du clavier et la conversion en codes de caractères utilisables.

Pour l'interrogation cyclique des touches, il utilise le mécanisme d'EVENT.

Voici les routines que nous avons sélectionnées:

KM WAIT CHAR va chercher un caractère dans le buffer clavier, dans la chaîne d'extension ou dans le buffer Put Back. Si aucun caractère n'est disponible, la routine ne revient pas. Elle attend obligatoirement. a contient s'il y a lieu le caractère qui a été entré au clavier.

KM READ CHAR transmet également un caractère dans a, s'il y en avait un, mais cette routine n'attend pas qu'il y ait un résultat positif. Si au retour de la routine, le carry est mis, c'est qu'il n'y avait pas de caractère à aller chercher.

Les routines KM WAIT KEY et KM READ KEY travaillent de façon similaire, mais seul le buffer clavier est interrogé. La chaîne d'extension et le buffer Put Back ne sont pas pris en compte.

KM SET REPEAT vous permet de déterminer quelles touches doivent être dotées de la fonction de répétition. Il faut placer en a le numéro de touche. b doit contenir &FF si la touche doit avoir une fonction de répétition et 0 s'il s'agit d'annuler la fonction de répétition de cette touche.

KEYBOARD MANAGER

```

***** KM INITIALISE
19E0 21021E      ld   hl,1E02
19E3 CD6D1C      call 1C6D      KM SET DELAY
19E6 AF          xor   a
19E7 320BB5      ld   (B50B),a
19EA 67          ld   h,a
19EB 6F          ld   l,a
19EC 22E7B4      ld   (B4E7),hl (Shift Lock State)
19EF 213CB4      ld   hl,B43C
19F2 11B0FF      ld   de,FFB0
19F5 2247B5      ld   (B547),hl (Adr. de table Repeat)
19F8 19          add  hl,de
19F9 2245B5      ld   (B545),hl (Adr. Key CTRL Table)
19FC 19          add  hl,de
19FD 2243B5      ld   (B543),hl (Adr. Key SHIFT Table)
1A00 19          add  hl,de
1A01 2241B5      ld   (B541),hl (Adr. Key Translation Table)
1A04 EB          ex   de,hl
1A05 21691D      ld   hl,1D69  Key Translation Table
1A08 01FA00      ld   bc,00FA
1A0B EDB0        ldir
1A0D 060A        ld   b,0A
1A0F 21EBB4      ld   hl,B4EB  Key State Map
1A12 3600        ld   (hl),00
1A14 23          inc  hl
1A15 10FB        djnz 1A12
1A17 060A        ld   b,0A
1A19 36FF        ld   (hl),FF
1A1B 23          inc  hl
1A1C 10FB        djnz 1A19

***** KM RESET
1A1E CDED1C      call 1CED
1A21 CD751A      call 1A75
1A24 1146B4      ld   de,B446
1A27 219800      ld   hl,0098
1A2A CD811A      call 1A81      KM EXP BUFFER CONT'D
1A2D 21361A      ld   hl,1A36  Restore KM Indirection
1A30 CD8A0A      call 0A8A      Move (hl+3)=>((hl+1)),cnt=(hl)
1A33 C3821C      jp   1C82      KM DISARM BREAK

```

KEYBOARD MANAGER

```

1A36 03          db    03          3 Octets
1A37 EEBC       dw    BDEE       Adresse objet
1A39 C32F1C     jp    1C2F       KM TEST BREAK

```

***** KM WAIT CHAR

```

1A3C CD421A     call 1A42       KM READ CHAR
1A3F
      30FB       jr    nc,1A3C   KM WAIT CHAR
1A41 C9         ret

```

***** KM READ CHAR

```

1A42 E5         push hl
1A43 21E0B4     ld    hl,B4E0   Put Back Buffer
1A46 7E         ld    a,(hl)    aller chercher caractère
1A47 36FF       ld    (hl),FF   vider buffer
1A49 BE         cp    (hl)      y avait-il un caractère ?
1A4A 3827       jr    c,1A73    oui =>
1A4C 2ADEB4     ld    hl,(B4DE) (Exp. String Pointer)
1A4F 7C         ld    a,h
1A50 B7         or    a         Exp. String ?
1A51 2011       jr    nz,1A64   oui =>
1A53 CD5C1B     call 1B5C       KM READ KEY
1A56 301B       jr    nc,1A73   aucun caractère =>
1A58 FE80       cp    80        Caractère < 128 ?
1A5A 3817       jr    c,1A73    oui =>
1A5C FEA0       cp    A0
1A5E 3F        ccf
1A5F 3812       jr    c,1A73
1A61 67         ld    h,a
1A62 2E00       ld    l,00
1A64 D5         push de
1A65 CD2E1B     call 1B2E       KM GET EXPAND
1A68 3802       jr    c,1A6C
1A6A 2600       ld    h,00
1A6C 2C         inc    l
1A6D 22DEB4     ld    (B4DE),hl (Exp. String Pointer)
1A70 D1         pop    de
1A71 30E0       jr    nc,1A53
1A73 E1         pop    hl
1A74 C9         ret

```

KEYBOARD MANAGER

```

1A75 3EFF          ld  a,FF
1A77 32E0B4       ld  (B4E0),a (Put Back Buffer)
1A7A C9           ret

1A7B CD811A       call 1A81      KM EXP BUFFER CONT'D
1A7E 3F           ccf
1A7F FB           ei
1A80 C9           ret

```

***** KM EXP BUFFER CONT'D

```

1A81 F3           di
1A82 7D           ld  a,l
1A83 D631         sub  31
1A85 7C           ld  a,h
1A86 DE00         sbc  a,00
1A88 D8           ret  c
1A89 19           add  hl,de
1A8A 22E3B4       ld  (B4E3),hl (Pointeur fin Exp Buffer)
1A8D EB           ex  de,hl
1A8E 22E1B4       ld  (B4E1),hl (Pointeur début Exp Buffer)
1A91 01300A       ld  bc,0A30  ASCII
1A94 3601         ld  (hl),01  0
1A96 23           inc  hl      à
1A97 71           ld  (hl),c  9
1A98 23           inc  hl      dans
1A99 0C           inc  c      Expansion
1A9A 10F8         djnz 1A94   Buffer
1A9C EB           ex  de,hl   Restore
1A9D 21B31A       ld  hl,1AB3 Default Exp String
1AA0 0E0A         ld  c,0A
1AA2 EDB0         ldir
1AA4 EB           ex  de,hl
1AA5 0613         ld  b,13
1AA7 AF           xor  a
1AA8 77           ld  (hl),a
1AA9 23           inc  hl
1AAA 10FC         djnz 1AA8
1AAC 22E5B4       ld  (B4E5),hl (Pointeur Exp Buffer libre)
1AAF 32DFB4       ld  (B4DF),a
1AB2 C9           ret

```


KEYBOARD MANAGER

***** Default Exp String

1AB3 01 2E 01 0D 05 52 55 4E RUN
1ABB 22 0D

***** KM SET EXPAND

```

1ABD F3          dl
1ABE 78          ld  a,b
1ABF CD3E1B      call 1B3E   Adr. Exp String => de
1AC2 301F        jr  nc,1AE3   Token non valable =>
1AC4 C5          push bc
1AC5 D5          push de
1AC6 E5          push hl
1AC7 CDE51A      call 1AE5   nettoyer Exp Buffer
1ACA 3F          ccf
1ACB E1          pop  hl
1ACC D1          pop  de
1ACD C1          pop  bc
1ACE 3013        jr  nc,1AE3
1AD0 1B          dec  de
1AD1 79          ld  a,c
1AD2 0C          inc  c
1AD3 12          ld  (de),a
1AD4 13          inc  de
1AD5 E7          rst  4      1d6a,(hl)
1AD6 23          inc  hl
1AD7 0D          dec  c
1AD8 20F9        jr  nz,1AD3
1ADA 21DFB4      ld  hl,B4DF
1ADD 78          ld  a,b
1ADE AE          xor  (hl)
1ADF 2001        jr  nz,1AE2
1AE1 77          ld  (hl),a
1AE2 37          scf
1AE3 FB          ei
1AE4 C9          ret
***** nettoyer Exp Buffer
1AE5 0600        ld  b,00
1AE7 60          ld  h,b
1AE8 6F          ld  l,a

```

KEYBOARD MANAGER

```

1AE9 79      ld  a,c
1AEA 95      sub  l
1AEB C8      ret  z
1AEC 300F    jr  nc,1AFD
1AEE 7D      ld  a,l
1AEF 69      ld  l,c
1AF0 4F      ld  c,a
1AF1 19      add  hl,de
1AF2 EB      ex  de,hl
1AF3 09      add  hl,bc
1AF4 CD221B  call 1B22      Place pour nouvelle Exp String?
1AF7 2823Jr  jr  z,1B1C      non =>
1AF9 EDB0    ldir
1AFB 181F    jr  1B1C
1AFD 4F      ld  c,a
1AFE 19      add  hl,de
1AFF E5      push hl
1B00 2AE5B4  ld  hl,(B4E5) (Pointeur Exp Buffer libre)
1B03 09      add  hl,bc
1B04 EB      ex  de,hl
1B05 2AE3B4  ld  hl,(B4E3) (Pointeur fin Exp Buffer)
1B08 7D      ld  a,l
1B09 93      sub  e
1B0A 7C      ld  a,h
1B0B 9A      sbc  a,d
1B0C E1      pop  hl
1B0D D8      ret  c
1B0E CD221B  call 1B22      Place pour nouvelle Exp String?
1B11 2AE5B4  ld  hl,(B4E5) (Pointeur Exp Buffer libre)
1B14 2806    jr  z,1B1C      Non =>
1B16 D5      push de
1B17 1B      dec  de
1B18 2B      dec  hl
1B19 EDB8    lddr
1B1B D1      pop  de
1B1C ED53E5B4 ld  (B4E5),de (Pointer Exp Buffer libre)
1B20 B7      or  a
1B21 C9      ret

```

***** place pour nouvelle Exp String?

KEYBOARD MANAGER

```

1B22 3AE5B4      ld  a,(B4E5)  (Pointer Exp Buffer 11bre)
1B25 95          sub  l
1B26 4F          ld  c,a
1B27 3AE6B4      ld  a,(B4E6)
1B2A 9C          sbc  a,h
1B2B 47          ld  b,a
1B2C B1          or   c
1B2D C9          ret

```

***** KM GET EXPAND

```

1B2E CD3E1B      call 1B3E      Adr. Exp String dans de
1B31 D0          ret  nc
1B32 BD          cp   l
1B33 C8          ret  z
1B34 3F          ccf
1B35 D0          ret  nc
1B36 E5          push hl
1B37 2600        ld  h,00
1B39 19          add  hl,de
1B3A 7E          ld  a,(hl)
1B3B E1          pop  hl
1B3C 37          scf
1B3D C9          ret

```

***** Adr. Exp String dans de

```

1B3E E67F        and  7F        Token dans zone valable?
1B40 FE20        cp   20
1B42 D0          ret  nc        non =>
1B43 E5          push hl
1B44 2AE1B4      ld  hl,(B4E1) (Pointer Start Exp Buffer)
1B47 110000      ld  de,0000
1B4A 3C          inc  a
1B4B 19          add  hl,de      ajouter à hl la longueur
1B4C 5E          ld  e,(hl)     de l'Expansion String
1B4D 23          inc  hl
1B4E 3D          dec  a
1B4F 20FA        jr   nz,1B4B
1B51 7B          ld  a,e
1B52 EB          ex  de,hl
1B53 E1          pop  hl

```

KEYBOARD MANAGER

1B54 37 scf
 1B55 C9 ret

***** KM WAIT KEY

1B56 CD5C1B call 1B5C KM READ KEY
 1B59 30FB Jr nc,1B56 KM WAIT KEY
 1B5B C9 ret

***** KM READ KEY

1B5C E5 push hl
 1B5D C5 push bc
 1B5E CD151D call 1D15
 1B61 303A Jr nc,1B9D
 1B63 79 ld a,c
 1B64 FEEF cp EF
 1B66 2834 Jr z,1B9C
 1B68 E60F and 0F
 1B6A 87 add a,a
 1B6B 87 add a,a
 1B6C 87 add a,a
 1B6D 3D dec a
 1B6E 3C inc a
 1B6F CB08 rrc b
 1B71 30FB Jr nc,1B6E
 1B73 CDA01B call 1BA0
 1B76 21E8B4 ld hl,B4E8 Caps Lock State
 1B79 CB7E bit 7,(hl)
 1B7B 280A Jr z,1B87
 1B7D FE61 cp 61
 1B7F 3806 Jr c,1B87
 1B81 FE7B cp 7B
 1B83 3002 Jr nc,1B87
 1B85 C6E0 add a,E0
 1B87 FEFF cp FF
 1B89 28D3 Jr z,1B5E
 1B8B FEFE cp FE
 1B8D 21E7B4 ld hl,B4E7 Shift Lock State
 1B90 2805 Jr z,1B97
 1B92 FEFD cp FD caps lock ?
 1B94 23 inc hl

KEYBOARD MANAGER

```

1B95 2005      Jr  nz,1B9C   non =>
1B97 7E        ld  a,(h1)
1B98 2F        cpl                toggle caps lock
1B99 77        ld  (h1),a
1B9A 18C2      Jr  1B5E
1B9C 37        scf
1B9D C1        pop  bc
1B9E E1        pop  hl
1B9F C9        ret

1BA0 CB11      rl  c
1BA2 DA481D   Jp  c,1D48   KM GET CONTROL
1BA5 47        ld  b,a
1BA6 3AE7B4   ld  a,(B4E7) (Shift Lock State)
1BA9 B1        or  c
1BAA E640      and 40
1BAC 78        ld  a,b
1BAD C2431D   Jp  nz,1D43  KM GET SHIFT
1BB0 C33E1D   Jp  1D3E    KM GET TRANSLATE

***** KM GET STATE
1BB3 2AE7B4   ld  hl,(B4E7) (Shift Lock State)
1BB6 C9        ret

***** Update Key State Map
1BB7 11FFB4   ld  de,B4FF  Multihit contr. à B4F5
1BBA 21F5B4   ld  hl,B4F5  Scan touches enfoncées
1BBD CD4608   call 0846    Scan Keyboard
1BC0 3A01B5   ld  a,(B501)
1BC3 E6A0     and  A0      isoler SHIFT/CTRL
1BC5 4F        ld  c,a
1BC6 21EDB4   ld  hl,B4ED  Key 16...23
1BC9 B6        or  (hl)
1BCA 77        ld  (hl),a
1BCB 21FFB4   ld  hl,B4FF  Multihit contr. à B4F5
1BCE 11EBB4   ld  de,B4EB  Key State Map
1BD1 0600     ld  b,00
1BD3 1A        ld  a,(de)
1BD4 AE        xor  (hl)
1BD5 A6        and  (hl)

```

KEYBOARD MANAGER

```

1BD6 C4481C      call  nz,1C48
1BD9 7E          ld    a,(hl)
1BDA 12          ld    (de),a
1BDB 23          inc   hl
1BDC 13          inc   de
1BDD 0C          inc   c
1BDE 79          ld    a,c
1BDF E60F        and   0F
1BE1 FE0A        cp    0A
1BE3 20EE        jr    nz,1BD3
1BE5 79          ld    a,c
1BE6 E6A0        and   A0
1BE8 CB71        bit   6,c
1BEA 4F          ld    c,a
1BEB C4EEBD      call  nz,BDEE    KM TEST BREAK
1BEE 78          ld    a,b
1BEF B7          or    a
1BF0 C0          ret   nz
1BF1 2109B5      ld    hl,B509
1BF4 35          dec   (hl)
1BF5 C0          ret   nz
1BF6 2A0AB5      ld    hl,(B50A)
1BF9 EB          ex   de,hl
1BFA 42          ld    b,d
1BFB 1600        ld    d,00
1BFD 21EBB4      ld    hl,B4EB    Key State Map
1C00 19          add   hl,de
1C01 7E          ld    a,(hl)
1C02 2A47B5      ld    hl,(B547) (Adr. de table Repeat)
1C05 19          add   hl,de
1C06 A6          and   (hl)
1C07 A0          and   b
1C08 C8          ret   z
1C09 2109B5      ld    hl,B509
1C0C 34          inc   (hl)
1COD 3A40B5      ld    a,(B540)
1C10 B7          or    a
1C11 C0          ret   nz
1C12 79          ld    a,c
1C13 B3          or    e

```

KEYBOARD MANAGER

```

1C14 4F          ld    c,a
1C15 3AE9B4     ld    a,(B4E9) (KM Delay)
1C18 3209B5     ld    (B509),a
1C1B CDFE1C     call 1CFE
1C1E 79         ld    a,c
1C1F E60F       and   0F
1C21 6F         ld    l,a
1C22 60         ld    h,b
1C23 220AB5     ld    (B50A),h1
1C26 FE08       cp    08
1C28 C0         ret   nz
1C29 CB60       bit   4,b
1C2B C0         ret   nz
1C2C CBF1       set  6,c
1C2E C9         ret

```

***** KM TEST BREAK

```

1C2F 21F3B4     ld    h1,B4F3
1C32 CB56       bit   2,(h1)
1C34 C8         ret   z
1C35 79         ld    a,c
1C36 EEA0       xor   A0
1C38 2056       jr    nz,1C90    KM BREAK EVENT
1C3A C5         push  bc
1C3B 23         inc   h1
1C3C 060A       ld    b,0A
1C3E 8E         adc  a,(h1)
1C3F 2B         dec  h1
1C40 10FC       djnz 1C3E
1C42 C1         pop   bc
1C43 FEA4       cp    A4
1C45 2049       jr    nz,1C90    KM BREAK EVENT
1C47 C7         rst   0
1C48 E5         push  h1
1C49 D5         push  de
1C4A 5F         ld    e,a
1C4B 2F         cpl
1C4C 3C         inc  a
1C4D A3         and  e
1C4E 47         ld    b,a

```

KEYBOARD MANAGER

```

1C4F 3AEAB4      ld   a,(B4EA)
1C52 CD181C      call 1C18
1C55 78          ld   a,b
1C56 AB          xor  e
1C57 20F1        jr   nz,1C4A
1C59 D1          pop  de
1C5A E1          pop  hl
1C5B C9          ret

```

***** KM GET JOYSTICK

```

1C5C 3AF1B4      ld   a,(B4F1) (Joystick 1)
1C5F E67F        and  7F
1C61 6F          ld   l,a
1C62 3AF4B4      ld   a,(B4F4) (Joystick 0)
1C65 E67F        and  7F
1C67 67          ld   h,a
1C68 C9          ret

```

***** KM GET DELAY

```

1C69 2AE9B4      ld   hl,(B4E9) (KM Delay)
1C6C C9          ret

```

***** KM SET DELAY

```

1C6D 22E9B4      ld   (B4E9),hl (KM Delay)
1C70 C9          ret

```

***** KM ARM BREAK

```

1C71 CD821C      call 1C82      KM DISARM BREAK
1C74 210DB5      ld   hl,B50D  Break Event Block
1C77 0640        ld   b,40
1C79 CDD201      call 01D2      KL INIT EVENT
1C7C 3EFF        ld   a,FF
1C7E 320CB5      ld   (B50C),a
1C81 C9          ret

```

***** KM DISARM BREAK

```

1C82 C5          push bc
1C83 D5          push de
1C84 210CB5      ld   hl,B50C

```


KEYBOARD MANAGER

```

1C87 3600      ld  (h1),00
1C89 23       inc  h1
1C8A CD8502   call 0285      KL DEL SYNCHRONOUS
1C8D D1       pop  de
1C8E C1       pop  bc
1C8F C9       ret

```

***** KM BREAK EVENT

```

1C90 210CB5   ld  h1,B50C
1C93 7E       ld  a,(h1)
1C94 3600     ld  (h1),00
1C96 BE       cp  (h1)
1C97 C8       ret  z
1C98 C5       push bc
1C99 D5       push de
1C9A 23       inc  h1
1C9B CDE201   call 01E2      KL EVENT
1C9E OEEF     ld  c,EF
1CA0 CDFE1C   call 1CFE
1CA3 D1       pop  de
1CA4 C1       pop  bc
1CA5 C9       ret

```

```

1CA6 2A47B5   ld  h1,(B547) (Adr. de table Repeat)
1CA9 181D     jr  1CC8      fixer Z suivant bit touche
1CAB FE50     cp  50       Key > 80 ?
1CAD D0       ret  nc      oui => non valable
1CAE 2A47B5   ld  h1,(B547) (Adr. de table Repeat)
1CB1 CDCD1C   call 1CCD     aller chercher bit corresp. touche #
1CB4 4F       ld  c,a
1CB5 2F       cpl
1CB6 A6       and  (h1)
1CB7 77       ld  (h1),a
1CB8 79       ld  a,c
1CB9 A0       and  b       (b=$ff/00)
1CBA B6       or   (h1)
1CBB 77       ld  (h1),a
1CBC C9       ret

```

***** KM TEST KEY

KEYBOARD MANAGER

```

1CBD F5          push af
1CBE 3AEDB4      ld a,(B4ED) (Key 16...23)
1CC1 E6A0        and A0          isoler SHIFT/CTRL
1CC3 4F          ld c,a
1CC4 F1          pop af
1CC5 21EBB4      ld hl,B4EB     Key State Map
1CC8 CDCD1C      call 1CCD      aller chercher bit corresp. touche #
1CCB A6          and (hl)       masquer bit touche
1CCC C9          ret

```

***** aller chercher bit corresp. touche #

```

1CCD D5          push de
1CCE F5          push af
1CCF E6F8        and F8          Key#
1CD1 0F          rrca          /8
1CD2 0F          rrca
1CD3 0F          rrca
1CD4 5F          ld e,a
1CD5 1600        ld d,00
1CD7 19          add hl,de      adresser Key Map
1CD8 F1          pop af
1CD9 E5          push hl
1CDA 21E51C      ld hl,1CE5     charger carte bits
1CDD E607        and 07         correspondant
1CDF 5F          ld e,a        à la touche
1CE0 19          add hl,de
1CE1 7E          ld a,(hl)
1CE2 E1          pop hl
1CE3 D1          pop de
1CE4 C9          ret

```

***** Cartes bits

```

1CE5 01 02 04 08 10 20 40 80

```

```

1CED F3          di
1CEE 213CB5      ld hl,B53C
1CF1 3615        ld (hl),15
1CF3 23          inc hl
1CF4 AF          xor a
1CF5 77          ld (hl),a

```

KEYBOARD MANAGER

```

1CF6 23      inc  h1
1CF7 3601    ld   (h1),01
1CF9 23      inc  h1
1CFA 77      ld   (h1),a
1CFB 23      inc  h1
1CFC 77      ld   (h1),a
1CFD C9      ret

1CFE 213CB5  ld   h1,B53C
1D01 B7      or   a
1D02 35      dec  (h1)
1D03 280E    jr   z,1D13
1D05 CD2C1D  call 1D2C
1D08 71      fd   (h1),c
1D09 23      inc  h1
1DOA 70      ld   (h1),b
1DOB 2140B5  ld   h1,B540
1DOE 34      inc  (h1)
1DOF 213EB5  ld   h1,B53E
1D12 37      scf
1D13 34      inc  (h1)
1D14 C9      ret

1D15 213EB5  ld   h1,B53E
1D18 B7      or   a
1D19 35      dec  (h1)
1D1A 280E    jr   z,1D2A
1D1C CD2C1D  call 1D2C
1D1F 4E      ld   c,(h1)
1D20 23      inc  h1
1D21 46      ld   b,(h1)
1D22 2140B5  ld   h1,B540
1D25 35      dec  (h1)
1D26 213CB5  ld   h1,B53C
1D29 37      scf
1D2A 34      inc  (h1)
1D2B C9      ret

1D2C 23      inc  h1
1D2D 34      inc  (h1)

```

KEYBOARD MANAGER

```

1D2E 7E      ld  a,(h1)
1D2F FE14    cp  14
1D31 2002    jr  nz,1D35
1D33 AF      xor  a
1D34 77      ld  (h1),a
1D35 87      add  a,a
1D36 CE14    adc  a,14
1D38 6F      ld  l,a
1D39 CEB5    adc  a,B5
1D3B 95      sub  l
1D3C 67      ld  h,a
1D3D C9      ret

```

***** KM GET TRANSLATE

```

1D3E 2A41B5  ld  h1,(B541) (Adr. Key Transl. Table)
1D41 1808    jr  1D4B      Get Key Table

```

***** KM GET SHIFT

```

1D43 2A43B5  ld  h1,(B543) (Adr. Key SHIFT Table)
1D46 1803    jr  1D4B      Get Key Table

```

ç

***** KM GET CONTROL

```

1D48 2A45B5  ld  h1,(B545) (Adr. Key CTRL Table)

```

***** Get Key Table

```

1D4B 85      add  a,l
1D4C 6F      ld  l,a
1D4D 8C      adc  a,h
1D4E 95      sub  l
1D4F 67      ld  h,a
1D50 7E      ld  a,(h1)
1D51 C9      ret

```

***** KM SET TRANSLATE

```

1D52 2A41B5  ld  h1,(B541) (Adr. Key Transl. Table)
1D55 1808    jr  1D5F      Set Key Table

```

***** KM SET SHIFT

```

1D57 2A43B5  ld  h1,(B543) (Adr. Key SHIFT Table)
1D5A 1803    jr  1D5F      Set Key Table

```

KEYBOARD MANAGER

***** KM SET CONTROL

1D5C 2A45B5 ld h1,(B545) (Adr. Key CTRL Table)

***** Set Key Table

1D5F FE50 cp 50
1D61 D0 ret nc
1D62 85 add a,l
1D63 6F ld l,a
1D64 8C adc a,h
1D65 95 sub l
1D66 67 ld h,a
1D67 70 ld (h1),b
1D68 C9 ret

***** Key Translation Table

1D69 F0 F3 F1 89 86 83 8B 8A
1D71 F2 E0 87 88 85 81 82 80
1D79 10 5B 0D 5D 84 FF 5C FF
1D81 5E 2D 40 70 3B 3A 2F 2E
1D89 30 39 6F 69 6C 6B 6D 2C
1D91 38 37 75 79 68 6A 6E 20
1D99 36 35 72 74 67 66 62 76
1DA1 34 33 65 77 73 64 63 78
1DA9 31 32 FC 71 09 61 FD 7A
1DB1 0B 0A 08 09 58 5A FF 7F

***** Key SHIFT Table

1DB9 F4 F7 F5 89 86 83 8B 8A
1DC1 F6 E0 87 88 85 81 82 80
1DC9 10 7B 0D 7D 84 FF 60 FF
1DD1 A3 3D 7C 50 2B 2A 3F 3E
1DD9 5F 29 4F 49 4C 4B 4D 3C
1DE1 28 27 55 59 48 4A 4E 20
1DE9 26 25 52 54 47 46 42 56
1DF1 24 23 45 57 53 44 43 58
1DF9 21 22 FC 51 09 41 FD 5A
1E01 0B 0A 08 09 58 5A FF 7F

***** Key CTRL Table

1E09 F8 FB F9 89 86 83 8C 8A

KEYBOARD MANAGER

```
1E11 FA E0 87 88 85 81 82 80
1E19 10 1B 0D 1D 84 FF 1C FF
1E21 1E FF 00 10 FF FF FF FF
1E29 1F FF 0F 09 0C 0B 0D FF
1E31 FF FF 15 19 08 0A 0E FF
1E39 FF FF 12 14 07 06 02 16
1E41 FF FF 05 17 13 04 03 18
1E49 FF 7E FC 11 E1 01 FE 1A
1E51 FF FF FF FF FF FF FF 7F
1E59 07 03 4B FF FF FF FF FF
1E61 AB 8F
```

```
1E63 C7          rst  0
1E64 C7          rst  0
1E65 C7          rst  0
1E66 C7          rst  0
1E67 C7          rst  0
```

SOUND MANAGER

2.5.8 SOUND MANAGER (SOUND)

Il n'y a pas grand chose à dire sur ce pack. La production du son proprement dite y prend en fait peu de place. La plus grande partie est occupée par la gestion des diverses files d'attente au rang desquelles figure également la réalisation de la TONE ENVELOPPE, que le PSG ne maîtrise pas de lui-même.

L'amateur de musique préférera sans doute programmer directement le PSG car les routines du SOUND sont trop taillées sur mesure pour les instructions Basic correspondantes. Pour jouer des mélodies, même à trois voix et même avec un tempo rapide, le Basic est très suffisant.

Pour le programmeur en langage-machine il serait tout au plus intéressant de réaliser une bonne percussion (c'est-à-dire avec des changements de son importants), ce qui n'est qu'imparfaitement possible en Basic avec des sons brefs mais complexes.

SOUND MANAGER

***** SOUND RESET

```

1E68 AF          xor    a
1E69 F3          di
1E6A 3252B5      ld     (B552),a (activité SOUND act.)
1E6D 3251B5      ld     (B551),a (ancienne act. SOUND (d'après HOLD))
1E70 2155B5      ld     hl,B555 Sound Event Block
1E73 11031F      ld     de,1F03 Sound Event
1E76 0681        ld     b,81
1E78 CDD201      call  01D2      KL INIT EVENT
1E7B 3E3F        ld     a,3F
1E7D 3219B6      ld     (B619),a
1E80 215CB5      ld     hl,B55C SOUND Params Canal A
1E83 013D00      ld     bc,003D
1E86 110801      ld     de,0108
1E89 AF          xor    a
1E8A 77          ld     (hl),a
1E8B 23          inc   hl
1E8C 72          ld     (hl),d
1E8D 23          inc   hl
1E8E 73          ld     (hl),e
1E8F 09          add   hl,bc
1E90 3C          inc   a
1E91 EB          ex    de,hl
1E92 29          add   hl,hl
1E93 EB          ex    de,hl
1E94 FE03        cp    03
1E96 20F2        jr    nz,1E8A
1E98 0E07        ld     c,07
1E9A DDE5        push  ix
1E9C E5          push  hl
1E9D 211DB5      ld     hl,B51D
1EA0 41          ld     b,c
1EA1 113F00      ld     de,003F
1EA4 19          add   hl,de
1EA5 CB38        srl   b
1EA7 30F8        jr    nc,1EA1
1EA9 C5          push  bc
1EAA E5          push  hl
1EAB DDE1        pop   ix
1EAD EB          ex    de,hl

```


SOUND MANAGER

```

1EAE CD7F22      call 227F
1EB1 13          inc  de
1EB2 13          inc  de
1EB3 13          inc  de
1EB4 6B          ld   l,e
1EB5 62          ld   h,d
1EB6 13          inc  de
1EB7 013B00     ld   bc,003B
1EBA 3600       ld   (hl),00
1EBC EDB0       ldir
1EBE DD361C04   ld   (ix+1C),04
1EC2 C1         pop  bc
1EC3 EB         ex   de,hl
1EC4 04         inc  b
1EC5 10DE       djnz 1EA5
1EC7 E1         pop  hl
1EC8 DDE1       pop  ix
1ECA C9         ret

```

***** SOUND HOLD

```

1ECB 2152B5     ld   hl,B552  Activité SOUND act.
1ECE F3         di
1ECF 7E         ld   a,(hl)
1ED0 3600       ld   (hl),00
1ED2 FB         ei
1ED3 B7         or   a        Canaux actifs ?
1ED4 C8         ret  z        Non =>
1ED5 2B         dec  hl
1ED6 77         ld   (hl),a
1ED7 2E03       ld   l,03    volume
1ED9 0E00       ld   c,00    de tous les canaux
1EDB 3E07       ld   a,07    sur 0
1EDD 85         add  a,l
1EDE CD2608     call 0826    MC SOUND REGISTER
1EE1 2D         dec  l
1EE2 20F7       jr   nz,1EDB
1EE4 37         scf
1EE5 C9         ret

```

***** SOUND CONTINUE

SOUND MANAGER

```

1EE6 3A51B5      ld  a,(B551)  (ancienne act. SOUND (d'après HOLD))
1EE9  B7         or  a         Canal actif ?
1EEA  C8         ret z        non =>
1EEB  DD211DB5   ld  ix,B51D
1EEF  113F00     ld  de,003F
1EF2  DD19       add  ix,de
1EF4  CB3F       srl  a         fixer ancien
1EF6  F5         push af      volume
1EF7  DD7E0F     ld  a,(ix+0F) pour tous les canaux
1EFA  DC7622     call c,2276
1EFD  F1         pop  af
1EFE  20F2       Jr  nz,1EF2
1F00  C31E20     jp  201E

```

***** Sound Event

```

1F03  DDE5       push ix
1F05  2150B5     ld  hl,B550
1F08  E5         push hl
1F09  AF         xor  a
1FOA  77         ld  (hl),a
1FOB  23         inc  hl
1FOC  46         ld  b,(hl)
1F0D  C5         push bc
1FOE  23         inc  hl      y a-t-il un
1F0F  B6         or  (hl)     canal actif ?
1F10  2822       Jr  z,1F34   non =>
1F12  DD211DB5   ld  ix,B51D
1F16  013F00     ld  bc,003F
1F19  DD09       add  ix,bc
1F1B  CB3F       srl  a         Canal actif ?
1F1D  30FA       Jr  nc,1F19  non => suivant
1F1F  F5         push af
1F20  DD7E04     ld  a,(ix+04)
1F23  1F         rra
1F24  DCC222     call c,22C2
1F27  DD7E07     ld  a,(ix+07)
1F2A  1F         rra
1F2B  DCB621     call c,21B6
1F2E  DCA820     call c,20A8
1F31  F1         pop  af

```

SOUND MANAGER

```

1F32 20E2      Jr   nz,1F16
1F34 C1        pop  bc
1F35 E1        pop  hl
1F36 7E        ld   a,(hl)
1F37 B7        or   a
1F38 2820      Jr   z,1F5A
1F3A 4F        ld   c,a
1F3B 23        inc  hl
1F3C 7E        ld   a,(hl)
1F3D 70        ld   (hl),b
1F3E A8        xor  b
1F3F 47        ld   b,a
1F40 23        inc  hl
1F41 B6        or   (hl)
1F42 77        ld   (hl),a
1F43 78        ld   a,b
1F44 2F        cpl
1F45 A1        and  c
1F46 2812      Jr   z,1F5A
1F48 DD211DB5  ld   ix,B51D
1F4C 113F00    ld   de,003F
1F4F DD19        add  ix,de
1F51 CB3F      srl  a
1F53 F5        push af
1F54 DC7F22    call c,227F
1F57 F1        pop  af
1F58 20F5      Jr   nz,1F4F
1F5A AF        xor  a
1F5B 3254B5    ld   (B554),a
1F5E DDE1      pop  ix
1F60 C9        ret

```

***** Scan Sound Queues

```

1F61 2152B5    ld   hl,B552  activité SOUND act.
1F64 7E        ld   a,(hl)
1F65 B7        or   a
1F66 C8        ret  z
1F67 23        inc  hl
1F68 35        dec  (hl)
1F69 C0        ret  nz

```

SOUND MANAGER

```

1F6A 34      inc  (hl)
1F6B 23      inc  hl
1F6C 7E      ld   a,(hl)
1F6D B7      or   a
1F6E C0      ret  nz
1F6F 2B      dec  hl
1F70 3603    ld   (hl),03
1F72 2B      dec  hl
1F73 46      ld   b,(hl)
1F74 2122B5  ld   hl,B522
1F77 113F00  ld   de,003F
1F7A AF      xor  a
1F7B 19      add  hl,de
1F7C CB38    srl  b
1F7E 30FB    jr   nc,1F7B
1F80 35      dec  (hl)
1F81 2005    jr   nz,1F88
1F83 2B      dec  hl
1F84 CB06    rlc  (hl)
1F86 8A      adc  a,d
1F87 23      inc  hl
1F88 23      inc  hl
1F89 35      dec  (hl)
1F8A 2005    jr   nz,1F91
1F8C 23      inc  hl
1F8D CB06    rlc  (hl)
1F8F 8A      adc  a,d
1F90 2B      dec  hl
1F91 2B      dec  hl
1F92 04      inc  b
1F93 10E6    djnz 1F7B
1F95 B7      or   a
1F96 C8      ret  z
1F97 2154B5  ld   hl,B554
1F9A 77      ld   (hl),a
1F9B 23      inc  hl
1F9C C3E201  jp   01E2      KL EVENT

```

***** SOUND QUEUE

```

1F9F CDE61E  call 1EE6      SOUND CONTINUE

```

SOUND MANAGER

```

1FA2 7E      ld  a,(hl)
1FA3 E607    and 07
1FA5 37      scf
1FA6 C8      ret  z
1FA7 4F      ld  c,a
1FA8 B6      or  (hl)
1FA9 FC9A1E  call m,1E9A.
1FAC 41      ld  b,c
1FAD DD211DB5 ld  ix,B51D
1FB1 113F00  ld  de,003F
1FB4 AF      xor  a
1FB5 DD19    add  ix,de
1FB7 CB38    srl  b
1FB9 30FA    jr  nc,1FB5
1FBB DD721E  ld  (ix+1E),d
1FBE DDBE1C  cp  (ix+1C)
1FC1 3F      ccf
1FC2 9F      sbc  a,a
1FC3 04      inc  b
1FC4 10EF    djnz 1FB5
1FC6 B7      or  a
1FC7 C0      ret  nz
1FC8 41      ld  b,c
1FC9 7E      ld  a,(hl)
1FCA 1F      rra
1FCB 1F      rra
1FCC 1F      rra
1FCD B0      or  b
1FCE E60F    and 0F
1FDO 4F      ld  c,a
1FD1 23      inc  hl
1FD2 DD211DB5 ld  ix,B51D
1FD6 113F00  ld  de,003F
1FD9 DD19    add  ix,de
1FDB CB38    srl  b
1FDD 30FA    jr  nc,1FD9
1FDF E5      push hl
1FE0 C5      push bc
1FE1 DD7E1B  ld  a,(ix+1B)
1FE4 DD341B  inc  (ix+1B)

```

SOUND MANAGER

1FE7	DD351C	dec	(ix+1C)	
1FEA	EB	ex	de,hl	
1FEB	CD3A20	call	203A	
1FEE	E5	push	hl	
1FEF	EB	ex	de,hl	
1FF0	DD7E01	ld	a,(ix+01)	
1FF3	2F	cpl		
1FF4	A1	and	c	
1FF5	12	ld	(de),a	
1FF6	13	inc	de	
1FF7	7E	ld	a,(hl)	
1FF8	23	inc	hl	
1FF9	87	add	a	
1FFA	87	add	a	
1FFB	87	add	a	
1FFC	87	add	a	
1FFD	47	ld	b,a	
1FFE	7E	ld	a,(hl)	
1FFF	23	inc	hl	
2000	E60F	and	0F	
2002	B0	or	b	
2003	12	ld	(de),a	
2004	13	inc	de	
2005	010600	ld	bc,0006	
2008	EDB0	ldir		
200A	E1	pop	hl	
200B	F3	di		
200C	DD7E1A	ld	a,(ix+1A)	
200F	DD341A	inc	(ix+1A)	
2012	DDB603	or	(ix+03)	
2015	FB	ei		
2016	CCBD20	call	z,20BD	
2019	C1	pop	bc	
201A	E1	pop	hl	
201B	04	inc	b	
201C	10B8	djnz	1FD6	
201E	E5	push	hl	
201F	2151B5	ld	hl,B551	ancienne act. SOUND (d'après HOLD)
2022	7E	ld	a,(hl)	
2023	B7	or	a	

SOUND MANAGER

```

2024 2811      Jr   z,2037
2026 3600      ld   (h1),00
2028 F3        dl   d1
2029 23        inc  h1
202A 46        ld   b,(h1)
202B B0        or   b
202C 77        ld   (h1),a
202D 78        ld   a,b
202E B7        or   a
202F 2005      Jr   nz,2036
2031 23        inc  h1
2032 3603      ld   (h1),03
2034 23        inc  h1
2035 77        ld   (h1),a
2036 FB        ei
2037 E1        pop  h1
2038 37        scf
2039 C9        ret

```

```

203A E603      and  03
203C 87        add  a,a
203D 87        add  a,a
203E 87        add  a,a
203F C61F     add  a,1F
2041 DDE5     push ix
2043 E1        pop  h1
2044 85        add  a,l
2045 6F        ld   l,a
2046 8C        adc  a,h
2047 95        sub  l
2048 67        ld   h,a
2049 C9        ret

```

***** SOUND RELEASE

```

204A 6F        ld   l,a
204B CDE61E    call 1EE6      SOUND CONTINUE
204E 7D        ld   a,l
204F E607      and  07
2051 C8        ret   z
2052 DD211DB5  ld   ix,B51D

```

SOUND MANAGER

```
2056 113F00      ld  de,003F
2059 DD19        add ix,de
205B CB3F        srl  a
205D 30FA       jr   nc,2059
205F F5         push af
2060 DDCB035E   bit  3,(ix+03)
2068 20EC       jr   nz,2056
206A 18B2       jr   201E
```

***** SOUND CHECK

```
206C E607        and  07
206E C8         ret  z
206F 2120B5     ld  hl,B520
2072 113F00     ld  de,003F
2075 19         add  hl,de
2076 1F         rra
2077 30FC       jr   nc,2075
2079 F3         di
207A 7E         ld  a,(hl)
207B 87         add  a,a
207C 87         add  a,a
207D 87         add  a,a
207E 111900     ld  de,0019
2081 19         add  hl,de
2082 B6         or   (hl)
2083 23         inc  hl
2084 23         inc  hl
2085 3600       ld  (hl),00
2087 FB        ei
2088 C9        ret
```

***** SOUND ARM EVENT

```
2089 E607        and  07
208B C8         ret  z
208C EB         ex  de,hl
208D 2139B5     ld  hl,B539
2090 013F00     ld  bc,003F
2093 09         add  hl,bc
2094 1F         rra
2095 30FC       jr   nc,2093
```


SOUND MANAGER

```

2097 AF      xor  a
2098 F3      di
2099 BE      cp   (hl)
209A 23      inc  hl
209B 73      ld  (hl),e
209C 23      inc  hl
209D 2003    jr   nz,20A2
209F 72      ld  (hl),d
20A0 FB      ei
20A1 C9      ret

20A2 77      ld  (hl),a
20A3 FB      ei
20A4 EB      ex  de,hl
20A5 C3E201  jp  01E2      KL EVENT

20A8 DD7E1A  ld  a,(ix+1A)
20AB B7      or  a
20AC CA7F22  jp  z,227F
20AF DD7E01  ld  a,(ix+01)
20B2 2150B5  ld  hl,B550
20B5 B6      or  (hl)
20B6 77      ld  (hl),a
20B7 DD7E19  ld  a,(ix+19)
20BA CD3A20  call 203A
20BD 7E      ld  a,(hl)
20BE B7      or  a
20BF 280C    jr   z,20CD
20C1 CB5F    bit  3,a
20C3 2053    jr   nz,2118
20C5 E5      push hl
20C6 3600    ld  (hl),00
20C8 CD1F21  call 211F
20CB E1      pop  hl
20CC D0      ret  nc
20CD DD360310  ld  (ix+03),10
20D1 23      inc  hl
20D2 7E      ld  a,(hl)
20D3 E6F0    and  F0
20D5 F5      push af

```

SOUND MANAGER

```

20D6 AE      xor    (h1)
20D7 5F      ld     e,a
20D8 23      inc   h1
20D9 4E      ld     c,(h1)
20DA 23      inc   h1
20DB 56      ld     d,(h1)
20DC 23      inc   h1
20DD B2      or    d
20DE B1      or    c
20DF 2808    jr    z,20E9
20E1 E5      push  h1
20E2 CDAB22  call  22AB
20E5 DD5601  ld     d,(ix+01)
20E8 E1      pop   h1
20E9 4E      ld     c,(h1)
20EA 23      inc   h1
20EB 5E      ld     e,(h1)
20EC 23      inc   h1
20ED 7E      ld     a,(h1)
20EE 23      inc   h1
20EF 66      ld     h,(h1)
20F0 6F      ld     l,a
20F1 F1      pop   af
20F2 CD7521  call  2175
20F5 2151B5  ld     hl,B551  Ancienne act. SOUND (d'après HOLD)
20F8 DD7E01  ld     a,(ix+01)
20FB B6      or    (h1)
20FC 77      ld     (hl),a
20FD DD3419  inc   (ix+19)
2100 DD351A  dec   (ix+1A)
2103 DD341C  inc   (ix+1C)
2106 F3      di
2107 DD7E1E  ld     a,(ix+1E)
210A DD361E00 ld     (ix+1E),00
210E FB      ei
210F B7      or    a
2110 C8      ret   z
2111 67      ld     h,a
2112 DD6E1D  ld     l,(ix+1D)
2115 C3E201  jp    01E2      KL EVENT

```

SOUND MANAGER

```

2118 CB9E      res    3,(hl)
211A DD360308   ld     (ix+03),08
211E C9         ret

211F DDE5      push  ix
2121 47        ld     b,a
2122 DD4E01     ld     c,(ix+01)
2125 DD215CB5  ld     ix,B55C   SOUND Params Canal A
2129 CB47     bit    0,a
212B 200C     jr     nz,2139
212D DD219BB5  ld     ix,B59B   SOUND Params Canal B
2131 CB4F     bit    1,a
2133 2004     jr     nz,2139
2135 DD21DAB5  ld     ix,B5DA   SOUND Params Canal C
2139 F3        di
213A DD7E03     ld     a,(ix+03)
213D A1        and   c
213E 282D     jr     z,216D
2140 78        ld     a,b
2141 DDBE01     cp    (ix+01)
2144 281A     jr     z,2160
2146 DDE5      push  ix
2148 DD21DAB5  ld     ix,B5DA   SOUND Params Canal C
214C CB57     bit    2,a
214E 2004     jr     nz,2154
2150 DD219BB5  ld     ix,B59B   SOUND Params Canal B
2154 DD7E03     ld     a,(ix+03)
2157 A1        and   c
2158 2812     jr     z,216C
215A FB        ei
215B CDB720     call  20B7
215E DDE1     pop   ix
2160 DD360300   ld     (ix+03),00
2164 FB        ei
2165 CDB720     call  20B7
2168 DDE1     pop   ix
216A 37        scf
216B C9        ret

216C E1        pop   hl

```

SOUND MANAGER

216D	DDE1	pop	1x	
216F	DD7003	ld	(1x+03),b	
2172	FB	ei		
2173	B7	or	a	
2174	C9	ret		
2175	CBFB	set	7,e	
2177	DD730F	ld	(1x+0F),e	
217A	5F	ld	e,a	
217B	7D	ld	a,l	
217C	B4	or	h	
217D	2001	Jr	nz,2180	
217F	2B	dec	hl	
2180	DD7508	ld	(1x+08),l	
2183	DD7409	ld	(1x+09),h	
2186	79	ld	a,c	
2187	B7	or	a	
2188	2808	Jr	z,2192	
218A	3E06	ld	a,06	charger générateur de bruit
218C	CD2608	call	0826	MC SOUND REGISTER
218F	DD7E02	ld	a,(1x+02)	
2192	B2	or	d	
2193	CD8B22	call	228B	
2196	7B	ld	a,e	
2197	B7	or	a	
2198	280A	Jr	z,21A4	
219A	210AB6	ld	hl,B60A	Courbes d'enveloppe de volume
219D	1600	ld	d,00	
219F	19	add	hl,de	
21A0	7E	ld	a,(hl)	
21A1	B7	or	a	
21A2	2003	Jr	nz,21A7	
21A4	21B221	ld	hl,21B2	
21A7	DD750A	ld	(1x+0A),l	
21AA	DD740B	ld	(1x+0B),h	
21AD	CD6522	call	2265	
21B0	180D	Jr	21BF	
21B2	010100	ld	bc,0001	
21B5	C8	ret	z	
21B6	DD6E0D	ld	l,(1x+0D)	

SOUND MANAGER

21B9	DD660E	ld	h, (1x+0E)	
21BC	DD5E10	ld	e, (1x+10)	
21BF	7B	ld	a, e	
21C0	FEFF	cp	FF	
21C2	2876	Jr	z, 223A	
21C4	87	add	a, a	
21C5	7E	ld	a, (h1)	
21C6	23	inc	h1	
21C7	384A	Jr	c, 2213	
21C9	280D	Jr	z, 21D8	
21CB	1D	dec	e	
21CC	B7	or	a	
21CD	2006	Jr	nz, 21D5	
21CF	DDB60F	or	(1x+0F)	
21D2	F2DD21	Jp	p, 21DD	
21D5	DD860F	add	a, (1x+0F)	
21D8	E60F	and	0F	
21DA	CD7322	call	2273	fixer volume
21DD	4E	ld	c, (h1)	
21DE	DD7E09	ld	a, (1x+09)	
21E1	47	ld	b, a	
21E2	87	add	a, a	
21E3	381B	Jr	c, 2200	
21E5	AF	xor	a	
21E6	91	sub	c	
21E7	DD8608	add	a, (1x+08)	
21EA	380C	Jr	c, 21F8	
21EC	05	dec	b	
21ED	F2F521	Jp	p, 21F5	
21F0	DD4E08	ld	c, (1x+08)	
21F3	AF	xor	a	
21F4	47	ld	b, a	
21F5	DD7009	ld	(1x+09), b	
21F8	DD7708	ld	(1x+08), a	
21FB	B0	or	b	
21FC	2002	Jr	nz, 2200	
21FE	1EFF	ld	e, FF	
2200	7B	ld	a, e	
2201	B7	or	a	
2202	CC4622	call	z, 2246	

SOUND MANAGER

2205	DD7310	ld	(1x+10),e	
2208	F3	di		
2209	DD7106	ld	(1x+06),c	
220C	DD360780	ld	(1x+07),80	
2210	FB	ei		
2211	B7	or	a	
2212	C9	ret		
2213	57	ld	d,a	
2214	4B	ld	c,e	
2215	3E0D	ld	a,0D	Courbe d'enveloppe
2217	CD2608	call	0826	MC SOUND REGISTER
221A	4A	ld	c,d	
221B	3E0B	ld	a,0B	longueur de courbe d'enveloppe Lo
221D	CD2608	call	0826	MC SOUND REGISTER
2220	4E	ld	c,(h1)	
2221	3E0C	ld	a,0C	longuer de courbe d'enveloppe H1
2223	CD2608	call	0826	MC SOUND REGISTER
2226	3E10	ld	a,10	
2228	CD7322	call	2273	fixer volume
222B	CD4622	call	2246	
222E	7B	ld	a,e	
222F	3C	inc	a	
2230	208D	jr	nz,21BF	
2232	21B221	ld	h1,21B2	
2235	CD6522	call	2265	
2238	1885	jr	21BF	
223A	AF	xor	a	
223B	DD7703	ld	(1x+03),a	
223E	DD7707	ld	(1x+07),a	
2241	DD7704	ld	(1x+04),a	
2244	37	scf		
2245	C9	ret		
2246	DD350C	dec	(1x+0C)	
2249	201E	jr	nz,2269	
224B	DD7E09	ld	a,(1x+09)	
224E	87	add	a,a	
224F	21B221	ld	h1,21B2	
2252	3011	jr	nc,2265	

SOUND MANAGER

```

2254 DD3408      inc   (ix+08)
2257 2006       Jr    nz,225F
2259 DD3409      inc   (ix+09)
225C 1EFF       ld    e,FF
225E C8          ret   z
225F DD6E0A      ld    l,(ix+0A)
2262 DD660B      ld    h,(ix+0B)
2265 7E         ld    a,(hl)
2266 DD770C      ld    (ix+0C),a
2269 23         inc   hl
226A 5E         ld    e,(hl)
226B 23         inc   hl
226C DD750D      ld    (ix+0D),l
226F DD740E      ld    (ix+0E),h
2272 C9         ret

```

***** fixer volume

```

2273 DD770F      ld    (ix+0F),a
2276 4F         ld    c,a
2277 DD7E00      ld    a,(ix+00)
227A C608       add   a,08      Volume
227C C32608      jp    0826     MC SOUND REGISTER

227F DD7E01      ld    a,(ix+01)
2282 2F         cpl
2283 2152B5     ld    hl,B552  activité SOUND act.
2286 F3         di
2287 A6         and   (hl)
2288 77         ld    (hl),a
2289 FB         ei
228A AF         xor   a
228B 47         ld    b,a
228C DD7E01      ld    a,(ix+01)
228F DDB602     or    (ix+02)
2292 2119B6     ld    hl,B619
2295 F3         di
2296 B6         or    (hl)
2297 A8         xor   b
2298 BE         cp    (hl)
2299 77         ld    (hl),a

```

SOUND MANAGER

229A	FB	ei	
229B	2003	Jr	nz,22A0
229D	78	ld	a,b
229E	B7	or	a
229F	C0	ret	nz
22A0	AF	xor	a
22A1	CD7622	call	2276
22A4	F3	di	
22A5	4E	ld	c,(h1)
22A6	3E07	ld	a,07
22A8	C32608	jp	0826
			Registre de commande - canal MC SOUND REGISTER
22AB	CD2423	call	2324
22AE	7B	ld	a,e
22AF	CD4E23	call	234E
22B2	D0	ret	nc
22B3	7E	ld	a,(h1)
22B4	E67F	and	7F
22B6	C8	ret	z
22B7	DD7511	ld	(1x+11),l
22BA	DD7412	ld	(1x+12),h
22BD	CD1323	call	2313
22C0	1809	Jr	22CB
22C2	DD6E14	ld	l,(1x+14)
22C5	DD6615	ld	h,(1x+15)
22C8	DD5E18	ld	e,(1x+18)
22CB	4E	ld	c,(h1)
22CC	23	inc	hl
22CD	7B	ld	a,e
22CE	D6F0	sub	F0
22D0	3804	Jr	c,22D6
22D2	1E00	ld	e,00
22D4	180E	Jr	22E4
22D6	1D	dec	e
22D7	79	ld	a,c
22D8	87	add	a,a
22D9	9F	sbc	a,a
22DA	57	ld	d,a
22DB	DD7E16	ld	a,(1x+16)
22DE	81	add	a,c
			SOUND T ADDRESS

SOUND MANAGER

22DF	4F	ld	c,a
22E0	DD7E17	ld	a,(ix+17)
22E3	8A	adc	a,d
22E4	57	ld	d,a
22E5	CD2423	call	2324
22E8	4E	ld	c,(hl)
22E9	7B	ld	a,e
22EA	B7	or	a
22EB	2019	jr	nz,2306
22ED	DD7E13	ld	a,(ix+13)
22F0	3D	dec	a
22F1	2010	jr	nz,2303
22F3	DD6E11	ld	l,(ix+11)
22F6	DD6612	ld	h,(ix+12)
22F9	7E	ld	a,(hl)
22FA	C680	add	a,80
22FC	3805	jr	c,2303
22FE	DD360400	ld	(ix+04),00
2302	C9	ret	
2303	CD1323	call	2313
2306	DD7318	ld	(ix+18),e
2309	F3	di	
230A	DD7105	ld	(ix+05),c
230D	DD360480	ld	(ix+04),80
2311	FB	ei	
2312	C9	ret	
2313	DD7713	ld	(ix+13),a
2316	23	inc	hl
2317	5E	ld	e,(hl)
2318	23	inc	hl
2319	DD7514	ld	(ix+14),l
231C	DD7415	ld	(ix+15),h
231F	7B	ld	a,e
2320	B7	or	a
2321	C0	ret	nz
2322	1C	inc	e
2323	C9	ret	
2324	DD7E00	ld	a,(ix+00)

SOUND MANAGER

```

2327 87      add  a,a
2328 F5      push af
2329 DD7116  ld   (ix+16),c Hauteur de ton Lo
232C CD2608  call 0826   MC SOUND REGISTER
232F F1      pop  af
2330 3C      inc  a
2331 4A      ld   c,d
2332 DD7117  ld   (ix+17),c Hauteur de ton HI
2335 C32608  jp   0826   MC SOUND REGISTER

```

***** SOUND AMPL ENVELOPE

```

2338 110AB6  ld   de,B60A Courbe d'enveloppe de volume
233B 1803    jr   2340   copier courbe d'enveloppe

```

***** SOUND TONE ENVELOPE

```

233D 11FAB6  ld   de,B6FA Courbes d'enveloppe de ton

```

***** copier courbe d'enveloppe

```

2340 EB      ex   de,hl
2341 CD5123  call 2351   aller chercher adr. courbe d'envel.
2344 EB      ex   de,hl
2345 D0      ret  nc
2346 EDB0    ldir
2348 C9      ret

```

***** SOUND A ADDRESS

```

2349 210AB6  ld   hl,B60A Courbes d'enveloppe de volume
234C 1803    jr   2351   aller chercher adr. courbe d'envel.

```

***** SOUND T ADDRESS

```

234E 21FAB6  ld   hl,B6FA Courbes d'enveloppe de ton

```

***** aller chercher adr. courbe d'envel.

```

2351 B7      or   a
2352 C8      ret  z
2353 FE10    cp   10
2355 D0      ret  nc
2356 011000  ld   bc,0010
2359 87      add  a,a
235A 87      add  a,a

```

SOUND MANAGER

235B	87	add	a,a
235C	87	add	a,a
235D	85	add	a,l
235E	6F	ld	l,a
235F	8C	adc	a,h
2360	95	sub	l
2361	67	ld	h,a
2362	37	scf	
2363	C9	ret	
2364	C7	rst	0
2365	C7	rst	0
2366	C7	rst	0
2367	C7	rst	0
2368	C7	rst	0
2369	C7	rst	0
236A	C7	rst	0
236B	C7	rst	0
236C	C7	rst	0
236D	C7	rst	0
236E	C7	rst	0
236F	C7	rst	0

2.5.9 CASSETTE MANAGER (CAS)

Le rôle de ce pack va de soi. L'utilisation des différentes routines ne présente pas de réel intérêt pour le programmeur en langage-machine car les programmes professionnels ne font pas en général bon ménage avec le lecteur de cassette. Le lecteur de disquette est en effet beaucoup plus satisfaisant dans ce cas.

Voici cependant quelques routines de base qui sont utilisables:

CAS IN OPEN ouvre un fichier d'entrée. Il faut pour cela placer en b la longueur du nom de fichier, en hl l'adresse de début du nom de fichier et en de l'adresse de début d'une zone de la Ram de 2 K qui sera utilisée comme buffer d'entrée.

Au retour de la routine, hl contient l'adresse de début de la tête de fichier (header).

a, bc et de contiennent d'autres valeurs tirées du header que vous pouvez cependant retirer vous-même directement du header, puisque vous disposez de l'adresse à laquelle il se trouve.

Les flags carry et zéro vous informent sur le succès de l'opération:

Carry=1 et zéro=0 signifient que tout a bien marché.

Carry=0 et zéro=0 signifient qu'il y a déjà un autre fichier d'ouvert.

Si la touche ESC a été enfoncée, carry=0 et zéro=1.

CAS OUT OPEN ouvre un fichier en sortie. Les paramètres à transmettre et la signification des flags sont les mêmes que ci-dessus. Naturellement, de doit ici contenir l'adresse du buffer de sortie.

CAS IN CHAR va chercher un caractère dans le buffer d'entrée et le transmet à travers a. Si c'était le dernier caractère du buffer, un nouveau bloc est automatiquement lu sur la cassette.

Si carry=0 et zéro=0, c'est que la fin du fichier (EOF) a été atteinte ou que le fichier n'était pas ouvert. Les autres combinaisons ont le même sens que ci-dessus.

CAS OUT CHAR écrit le caractère qui se trouve dans a dans le buffer de sortie. Si celui-ci est plein, il est automatiquement copié sur la cassette.

La signification des flags est la même que ci-dessus.

CASSETTE MANAGER

*****CAS INITIALISE

```

2370 CD0124      call 2401      CAS IN ABANDON
2373 CD2E24      call 242E      CAS OUT ABANDON
2376 AF          xor a
2377 CD8E23      call 238E      CAS NOISY
237A 214D01      ld hl,014D
237D 3E19        ld a,19

```

*****CAS SET SPEED

```

237F 29          add hl,hl
2380 29          add hl,hl
2381 29          add hl,hl
2382 29          add hl,hl
2383 29          add hl,hl
2384 29          add hl,hl
2385 0F          rrca
2386 0F          rrca
2387 E63F        and 3F
2389 6F          ld l,a
238A 22D1B8      ld (B8D1),hl  (Cass. Speed)
238D C9          ret

```

*****CAS NOISY

```

238E 3200B8      ld (B800),a  (Cass. Message Flag)
2391 C9          ret

```

*****CAS IN OPEN

```

2392 DD2102B8    ld 1x,B802  Input Buffer Status
2396 CDAF23      call 23AF   CAS Open
2399 D0          ret nc
239A E5          push hl
239B CD3F25      call 253F   lire header fichier
239E ED5B1CB8    ld de,(B81C)
23A2 ED4B1FB8    ld bc,(B81F)
23A6 3A19B8      ld a,(B819)
23A9 E1          pop hl
23AA C9          ret

```

*****CAS OUT OPEN

```

23AB DD2147B8    ld 1x,B847  Output Buffer Status

```

CASSETTE MANAGER

*****CAS Open

```

23AF DD7E00      ld  a,(1x+00)
23B2 B7         or  a
23B3 C0         ret nz
23B4 DDE5      push 1x
23B6 E3        ex  (sp),hl
23B7 3601      ld  (hl),01
23B9 23        inc  hl
23BA 73        ld  (hl),e
23BB 23        inc  hl
23BC 72        ld  (hl),d
23BD 23        inc  hl
23BE 73        ld  (hl),e
23BF 23        inc  hl
23C0 72        ld  (hl),d
23C1 23        inc  hl
23C2 EB        ex  de,hl
23C3 E1        pop  hl
23C4 D5        push de
23C5 0E40      ld  c,40
23C7 12        ld  (de),a
23C8 13        inc  de
23C9 0D        dec  c
23CA 20FB      jr  nz,23C7
23CC D1        pop  de
23CD D5        push de
23CE 78        ld  a,b
23CF FE10      cp  10
23D1 3802      jr  c,23D5
23D3 0610      ld  b,10
23D5 04        inc  b
23D6 48        ld  c,b
23D7 1807      jr  23E0
23D9 E7        rst  4
23DA 23        inc  hl
23DB CDB627    call 27B6
23DE 12        ld  (de),a
23DF 13        inc  de
23E0 10F7      djnz 23D9
23E2 0D        dec  c

```

CASSETTE MANAGER

```

23E3 2809      Jr    z,23EE
23E5 1B        dec   de
23E6 1A        ld    a,(de)
23E7 EE20     xor   20
23E9 2003     Jr    nz,23EE
23EB 12        ld    (de),a
23EC 18F4     Jr    23E2
23EE E1        pop   hl
23EF DD361501  ld    (1x+15),01
23F3 DD361716  ld    (1x+17),16
23F7 DD351C     dec   (1x+1C)
23FA 37        scf
23FB C9        ret

```

*****CAS IN CLOSE

```

23FC 3A02B8     ld    a,(B802)    (Input Buffer Status)
23FF B7         or    a
2400 C8        ret    z

```

*****CAS IN ABANDON

```

2401 2102B8     ld    hl,B802    Input Buffer Status
2404 3E01     ld    a,01
2406 3600     ld    (hl),00
2408 23       inc   hl
2409 5E       ld    e,(hl)
240A 23       inc   hl
240B 56       ld    d,(hl)
240C 21CCB8    ld    hl,B8CC
240F AE       xor   (hl)
2410 37       scf
2411 C0       ret   nz
2412 77       ld    (hl),a
2413 9F       sbc   a,a
2414 C9       ret

```

*****CAS OUT CLOSE

```

2415 3A47B8     ld    a,(B847)    (Output Buffer Status)
2418 FE04     cp    04
241A 2812     Jr    z,242E    CAS OUT ABANDON
241C C6FF     add   a,FF

```

CASSETTE MANAGER

```

241E D0          ret  nc
241F 215DB8     ld   hl,B85D
2422 36FF      ld   (hl),FF
2424 23        inc  hl
2425 23        inc  hl
2426 7E        ld   a,(hl)
2427 23        inc  hl
2428 B6        or   (hl)
2429 37        scf
242A C41426    call nz,2614
242D D0        ret  nc
  
```

*****CAS OUT ABANDON

```

242E 2147B8     ld   hl,B847      Output Buffer Status
2431 3E02      ld   a,02
2433 18D1      jr   2406
  
```

*****CAS IN CHAR

```

2435 E5        push hl
2436 D5        push de
2437 C5        push bc
2438 0602      ld   b,02
243A CD8B24    call 248B      Check Input Buffer Status
243D 201A      jr   nz,2459
243F 2A1AB8    ld   hl,(B81A)
2442 7C        ld   a,h
2443 B5        or   l
2444 37        scf
2445 CC3F25    call z,253F    lire header fichier
2448 300F      jr   nc,2459
244A 2A1AB8    ld   hl,(B81A)
244D 2B        dec  hl
244E 221AB8    ld   (B81A),hl
2451 2A05B8    ld   hl,(B805) (Pointer Input Buffer)
2454 E7        rst  4        ld a,(hl)
2455 23        inc  hl
2456 2205B8    ld   (B805),hl (Pointer Input Buffer)
2459 182C      jr   2487
  
```

*****CAS OUT CHAR

CASSETTE MANAGER

```

245B E5      push  hl
245C D5      push  de
245D C5      push  bc
245E 4F      ld    c,a
245F 2147B8  ld    hl,B847    Output Buffer Status
2462 0602    ld    b,02
2464 CD8E24  call  248E    Check Buffer Status
2467 201E    jr    nz,2487
2469 2A5FB8    ld    hl,(B85F)
246C 110008    ld    de,0800
246F ED52    sbc   hl,de
2471 C5      push  bc
2472 D41426    call  nc,2614
2475 C1      pop   bc
2476 300F    jr    nc,2487
2478 2A5FB8    ld    hl,(B85F)
247B 23      inc   hl
247C 225FB8    ld    (B85F),hl
247F 2A4AB8    ld    hl,(B84A)  (Pointer Output Buffer)
2482 71      ld    (hl),c
2483 23      inc   hl
2484 224AB8    ld    (B84A),hl  (Pointer Output Buffer)
2487 C1      pop   bc
2488 D1      pop   de
2489 E1      pop   hl
248A C9      ret

```

*****Check Input Buffer Status

```

248B 2102B8    ld    hl,B802    Input Buffer Status

```

*****Check Buffer Status

```

248E 7E      ld    a,(hl)
248F B8      cp    b
2490 C8      ret   z
2491 EE01    xor   01
2493 C0      ret   nz
2494 70      ld    (hl),b
2495 C9      ret

```

*****CAS TEST EOF

CASSETTE MANAGER

2496 CD3524 call 2435 CAS IN CHAR
 2499 D0 ret nc

*****CAS RETURN

249A E5 push hl
 249B 2A1AB8 ld hl,(B81A)
 249E 23 inc hl
 249F 221AB8 ld (B81A),hl
 24A2 2A05B8 ld hl,(B805) (Pointer Input Buffer)
 24A5 2B dec hl
 24A6 2205B8 ld (B805),hl (Pointer Input Buffer)
 24A9 E1 pop hl
 24AA C9 ret

*****CAS IN DIRECT

24AB EB ex de,hl
 24AC 0603 ld b,03
 24AE CD8B24 call 248B Check Input Buffer Status
 24B1 C0 ret nz
 24B2 ED531CB8 ld (B81C),de
 24B6 CDCF24 call 24CF
 24B9 2A1CB8 ld hl,(B81C)
 24BC ED5B1AB8 ld de,(B81A)
 24C0 19 add hl,de
 24C1 221CB8 ld (B81C),hl
 24C4 CD3F25 call 253F Lire header fichier
 24C7 38F0 Jr c,24B9
 24C9 C8 ret z
 24CA 2AA6B8 ld hl,(B8A6)
 24CD 37 scf
 24CE C9 ret

24CF 2A03B8 ld hl,(B803) (Adr. Start Input Buffer)
 24D2 ED5B1CB8 ld de,(B81C)
 24D6 ED4B1AB8 ld bc,(B81A)
 24DA 7B ld a,e
 24DB 95 sub l
 24DC 7A ld a,d
 24DD 9C sbc a,h
 24DE DAA6BA Jp c,BAA6 (0537)KL LDIR CONT'D
 24E1 09 add hl,bc

CASSETTE MANAGER

```

24E2 2B      dec  h1
24E3 EB      ex   de,h1
24E4 09      add  h1,bc
24E5 2B      dec  h1
24E6 EB      ex   de,h1
24E7 C3ACBA  jp   BAAC (053D) KL LDDR CONT'D

```

*****CAS OUT DIRECT

```

24EA E5      push h1
24EB C5      push bc
24EC 4F      ld   c,a
24ED 2147B8  ld   h1,B847      Output Buffer Status
24F0 0603    ld   b,03
24F2 CD8E24  call 248E      Check Buffer Status
24F5 79      ld   a,c
24F6 C1      pop  bc
24F7 E1      pop  h1
24F8 C0      ret  nz
24F9 325EB8  ld   (B85E),a
24FC ED5364B8 ld   (B864),de
2500 ED4366B8 ld   (B866),bc
2504 2248B8  ld   (B848),h1      (Adr. Start Output Buffer)
2507 ED535FB8 ld   (B85F),de
250B 21FFF7  ld   h1,F7FF
250E 19      add  h1,de
250F 3F      ccf
2510 D8      ret  c
2511 210008  ld   h1,0800
2514 225FB8  ld   (B85F),h1
2517 EB      ex   de,h1
2518 ED52    sbc  h1,de
251A E5      push h1
251B 2A48B8  ld   h1,(B848)      (Adr. Start Output Buffer)
251E 19      add  h1,de
251F E5      push h1
2520 CD1426  call 2614
2523 E1      pop  h1
2524 D1      pop  de
2525 D0      ret  nc
2526 * 18DC jr   2504

```

CASSETTE MANAGER

*****CAS CATALOG

2528	2102B8	ld	h1,B802	Input Buffer Status
252B	7E	ld	a,(h1)	
252C	B7	or	a	
252D	C0	ret	nz	
252E	3605	ld	(h1),05	
2530	ED5303B8	ld	(B803),de	(Adr. Start Input Buffer)
2534	CD8E23	call	238E	CAS NOISY
2537	CD4425	call	2544	
253A	38FB	Jr	c,2537	
253C	C30124	Jp	2401	CAS IN ABANDON

*****Lire header fichier

253F	3A18B8	ld	a,(B818)	
2542	B7	or	a	
2543	C0	ret	nz	
2544	010183	ld	bc,8301	
2547	CD7326	call	2673	
254A	305C	Jr	nc,25A8	
254C	218CB8	ld	h1,B88C	
254F	114000	ld	de,0040	
2552	3E2C	ld	a,2C	
2554	CD3628	call	2836	CAS READ
2557	304F	Jr	nc,25A8	
2559	CDC525	call	25C5	
255C	2057	Jr	nz,25B5	
255E	068B	ld	b,8B	
2560	3802	Jr	c,2564	
2562	0689	ld	b,89	
2564	CD9226	call	2692	
2567	ED5B9FB8	ld	de,(B89F)	
256B	2A1CB8	ld	h1,(B81C)	
256E	3A02B8	ld	a,(B802)	(Input Buffer Status)
2571	FE03	cp	03	
2573	280E	Jr	z,2583	
2575	21FFF7	ld	h1,F7FF	
2578	19	add	h1,de	
2579	3E04	ld	a,04	
257B	382B	Jr	c,25A8	
257D	2A03B8	ld	h1,(B803)	(Adr. Start Input Buffer)

CASSETTE MANAGER

2580	2205B8	ld	(B805),h1	(Pointer Input Buffer)
2583	3E16	ld	a,16	
2585	CD3628	call	2836	CAS READ
2588	301E	jr	nc,25A8	
258A	2117B8	ld	h1,B817	
258D	34	inc	(h1)	
258E	3A9DB8	ld	a,(B89D)	
2591	23	inc	h1	
2592	77	ld	(h1),a	
2593	AF	xor	a	
2594	321EB8	ld	(B81E),a	
2597	2A9FB8	ld	h1,(B89F)	
259A	221AB8	ld	(B81A),h1	
259D	CDBF27	call	27BF	
25A0	3E8C	ld	a,8C	
25A2	CC0C27	call	z,270C	
25A5	37	scf		
25A6	1865	jr	260D	
25A8	B7	or	a	
25A9	2102B8	ld	h1,B802	Input Buffer Status
25AC	285D	jr	z,260B	
25AE	0685	ld	b,85	
25B0	CD1327	call	2713	
25B3	1897	jr	254C	
25B5	F5	push	af	
25B6	0688	ld	b,88	
25B8	CD9226	call	2692	
25BB	F1	pop	af	
25BC	308E	jr	nc,254C	
25BE	0687	ld	b,87	
25C0	CD1127	call	2711	
25C3	1887	jr	254C	
25C5	CDBF27	call	27BF	
25C8	37	scf		
25C9	C8	ret	z	
25CA	3A1EB8	ld	a,(B81E)	
25CD	B7	or	a	
25CE	281B	jr	z,25EB	
25D0	3AA3B8	ld	a,(B8A3)	
25D3	2F	cpl		

CASSETTE MANAGER

25D4	B7	or	a	
25D5	C0	ret	nz	
25D6	3A07B8	ld	a,(B807)	(File Header Input)
25D9	B7	or	a	
25DA	C4F325	call	nz,25F3	
25DD	C0	ret	nz	
25DE	218CB8	ld	hl,B88C	
25E1	1107B8	ld	de,B807	File Header Input
25E4	014000	ld	bc,0040	
25E7	EDB0	ldir		
25E9	AF	xor	a	
25EA	C9	ret		
25EB	CDF325	call	25F3	
25EE	C0	ret	nz	
25EF	EB	ex	de,hl	
25F0	1A	ld	a,(de)	
25F1	BE	cp	(hl)	
25F2	C9	ret		
25F3	2107B8	ld	hl,B807	File Header Input
25F6	118CB8	ld	de,B88C	
25F9	0610	ld	b,10	
25FB	1A	ld	a,(de)	
25FC	CDB627	call	27B6	
25FF	4F	ld	c,a	
2600	7E	ld	a,(hl)	
2601	CDB627	call	27B6	
2604	A9	xor	c	
2605	C0	ret	nz	
2606	23	inc	hl	
2607	13	inc	de	
2608	10F1	djnz	25FB	
260A	C9	ret		
260B	3604	ld	(hl),04	
260D	9F	sbc	a,a	
260E	F5	push	af	
260F	CD4F2A	call	2A4F	CAS STOP MOTOR
2612	F1	pop	af	

CASSETTE MANAGER

2613	C9	ret	
2614	010284	ld	bc,8402
2617	CD7326	call	2673
261A	304A	jr	nc,2666
261C	068A	ld	b,8A
261E	114CB8	ld	de,B84C File Header Output
2621	CD9526	call	2695
2624	2163B8	ld	hl,B863
2627	CD8826	call	2688
262A	303A	jr	nc,2666
262C	2A48B8	ld	hl,(B848) (Adr. Start Output Buffer)
262F	224AB8	ld	(B84A),hl (Pointer Output Buffer)
2632	2261B8	ld	(B861),hl
2635	E5	push	hl
2636	214CB8	ld	hl,B84C File Header Output
2639	114000	ld	de,0040
263C	3E2C	ld	a,2C
263E	CD3F28	call	283F CAS WRITE
2641	E1	pop	hl
2642	3022	jr	nc,2666
2644	ED5B5FB8	ld	de,(B85F)
2648	3E16	ld	a,16
264A	CD3F28	call	283F CAS WRITE
264D	215DB8	ld	hl,B85D
2650	DC8826	call	c,2688
2653	3011	jr	nc,2666
2655	210000	ld	hl,0000
2658	225FB8	ld	(B85F),hl
265B	215CB8	ld	hl,B85C
265E	34	inc	(hl)
265F	AF	xor	a
2660	3263B8	ld	(B863),a
2663	37	scf	
2664	18A7	jr	260D
2666	B7	or	a
2667	2147B8	ld	hl,B847 Output Buffer Status
266A	289F	jr	z,260B
266C	0686	ld	b,86
266E	CD1327	call	2713

CASSETTE MANAGER

2671	18B9	Jr	262C	
2673	21CCB8	ld	hl,B8CC	
2676	79	ld	a,c	
2677	BE	cp	(hl)	
2678	3600	ld	(hl),00	
267A	37	scf		
267B	E5	push	hl	
267C	C5	push	bc	
267D	C46027	call	nz,2760	
2680	C1	pop	bc	
2681	E1	pop	hl	
2682	9F	sbc	a,a	
2683	D0	ret	nc	
2684	71	ld	(hl),c	
2685	C34B2A	Jp	2A4B	CAS START MOTOR
2688	7E	ld	a,(hl)	
2689	B7	or	a	
268A	37	scf		
268B	C8	ret	z	
268C	012C01	ld	bc,012C	
268F	C3722A	Jp	2A72	
2692	118CB8	ld	de,B88C	
2695	3A00B8	ld	a,(B800)	(Cass. Message Flag)
2698	B7	or	a	
2699	C0	ret	nz	
269A	3201B8	ld	(B801),a	
269D	CD8327	call	2783	
26A0	CD2627	call	2726	
26A3	1A	ld	a,(de)	
26A4	B7	or	a	
26A5	200A	Jr	nz,26B1	
26A7	3E8E	ld	a,8E	
26A9	CD2727	call	2727	
26AC	011000	ld	bc,0010	
26AF	182E	Jr	26DF	
26B1	CDBF27	call	27BF	
26B4	010010	ld	bc,1000	
26B7	280D	Jr	z,26C6	

CASSETTE MANAGER

26B9	6B	ld	l,e	
26BA	62	ld	h,d	
26BB	7E	ld	a,(hl)	
26BC	B7	or	a	
26BD	2804	jr	z,26C3	
26BF	0C	inc	c	
26C0	23	inc	hl	
26C1	10F8	djnz	26BB	
26C3	78	ld	a,b	
26C4	41	ld	b,c	
26C5	4F	ld	c,a	
26C6	CD8D27	call	278D	
26C9	1A	ld	a,(de)	
26CA	CDB627	call	27B6	
26CD	B7	or	a	
26CE	2002	jr	nz,26D2	
26D0	3E20	ld	a,20	
26D2	C5	push	bc	
26D3	D5	push	de	
26D4	CD3413	call	1334	TXT WR CHAR
26D7	D1	pop	de	
26D8	C1	pop	bc	
26D9	13	inc	de	
26DA	10ED	djnz	26C9	
26DC	CD5C27	call	275C	
26DF	EB	ex	de,hl	
26E0	09	add	hl,bc	
26E1	EB	ex	de,hl	
26E2	3E8D	ld	a,8D	
26E4	CD2727	call	2727	
26E7	0602	ld	b,02	
26E9	CD8D27	call	278D	
26EC	1A	ld	a,(de)	
26ED	CDA427	call	27A4	
26F0	CD5C27	call	275C	
26F3	13	inc	de	
26F4	CDBF27	call	27BF	
26F7	200B	jr	nz,2704	
26F9	13	inc	de	
26FA	1A	ld	a,(de)	

CASSETTE MANAGER

```

26FB E60F      and  0F
26FD C624      add  a,24
26FF CD8027    call 2780      sortir message CAS (1 caract.)
2702 1858      jr   275C
2704 1A        ld   a,(de)
2705 2101B8    ld   hl,B801
2708 B6        or   (hl)
2709 C8        ret  z
270A 186F      jr   277B
270C CD2727    call 2727
270F 186A      jr   277B
2711 3EFF      ld   a,FF
2713 F5        push af
2714 CD1F27    call 271F      sortir message CAS (# dans b)
2717 F1        pop  af
2718 C660      add  a,60
271A D48027    call nc,2780   sortir message CAS (1 caractère)
271D 185C      jr   277B

```

*****sortir message CAS (# dans b)

```

271F CD8011    call 1180      TXT GET CURSOR
2722 25        dec  h
2723 C47B27    call nz,277B
2726 78        ld   a,b
2727 E5        push hl
2728 E67F      and  7F
272A 47        ld   b,a
272B 21C527    ld   hl,27C5   messages cassette
272E 2807      jr   z,2737
2730 7E        ld   a,(hl)
2731 23        inc  hl
2732 B7        or   a
2733 20FB      jr   nz,2730
2735 10F9      djnz 2730
2737 7E        ld   a,(hl)
2738 B7        or   a
2739 2805      jr   z,2740
273B CD4327    call 2743
273E 18F7      jr   2737
2740 E1        pop  hl

```

CASSETTE MANAGER

2741	23	inc	hl	
2742	C9	ret		
2743	FA2727	jp	m,2727	
2746	E5	push	hl	
2747	0600	ld	b,00	
2749	04	inc	b	
274A	7E	ld	a,(hl)	
274B	23	inc	hl	
274C	07	rlca		
274D	30FA	jr	nc,2749	
274F	CD8D27	call	278D	
2752	E1	pop	hl	
2753	7E	ld	a,(hl)	
2754	23	inc	hl	
2755	E67F	and	7F	
2757	CD8027	call	2780	sortir message CAS (1 caractère)
275A	10F7	djnz	2753	
275C	3E20	ld	a,20	
275E	1820	jr	2780	sortir message CAS (1 Caractère)
2760	3A00B8	ld	a,(B800)	(Cass. Message Flag)
2763	B7	or	a	
2764	37	scf		
2765	C0	ret	nz	
2766	CD1F27	call	271F	sortir message CAS (# dans b)
2769	CD421A	call	1A42	KM READ CHAR
276C	38FB	jr	c,2769	
276E	CD7912	call	1279	TXT CUR ON
2771	CD561B	call	1B56	KM WAIT KEY
2774	CD8112	call	1281	TXT CUR OFF
2777	FE1B	cp	1B	
2779	C8	ret	z	
277A	37	scf		
277B	CD8327	call	2783	
277E	3E0A	ld	a,0A	
*****sortir message CAS (1 caractère)				
2780	C30014	jp	1400	TXT OUTPUT
2783	F5	push	af	

CASSETTE MANAGER

2784	E5	push	hl	
2785	3E01	ld	a,01	
2787	CD5E11	call	115E	TXT SET COLUMN
278A	E1	pop	hl	
278B	F1	pop	af	
278C	C9	ret		
278D	D5	push	de	
278E	CD5612	call	1256	TXT GET WINDOW
2791	5C	ld	e,h	
2792	CD8011	call	1180	TXT GET CURSOR
2795	7C	ld	a,h	
2796	3D	dec	a	
2797	83	add	a,e	
2798	80	add	a,b	
2799	3D	dec	a	
279A	BA	cp	d	
279B	D1	pop	de	
279C	D8	ret	c	
279D	3EFF	ld	a,FF	
279F	3201B8	ld	(B801),a	
27A2	18D7	Jr	277B	
27A4	06FF	ld	b,FF	
27A6	04	inc	b	
27A7	D60A	sub	0A	
27A9	30FB	Jr	nc,27A6	
27AB	C63A	add	a,3A	
27AD	F5	push	af	
27AE	78	ld	a,b	
27AF	B7	or	a	
27B0	C4A427	call	nz,27A4	
27B3	F1	pop	af	
27B4	18CA	Jr	2780	sortir message CAS (1 caractère)
27B6	FE61	cp	61	
27B8	D8	ret	c	
27B9	FE7B	cp	7B	
27BB	D0	ret	nc	
27BC	C6E0	add	a,E0	
27BE	C9	ret		

CASSETTE MANAGER

```
27BF 3A02B8      ld  a,(B802)    (Input Buffer Status)
27C2 FE05        cp   05
27C4 C9          ret
```

*****messages cassette

```
27C5 50 72 65 73 F3 00 50 4C      Press.PL
27CD 41 D9 74 68 65 EE 61 6E      AYthenan
27D5 F9 6B 65 79 BA 00 65 72     ykey:.er
27DD 72 6F F2 00 80 81 00 80     ror....
27E5 52 45 C3 61 6E E4 81 00     RECand..
27ED 52 65 61 E4 82 00 57 72     Read..Wr
27F5 69 74 E5 82 00 52 65 77     ite..rew
27FD 69 6E E4 74 61 70 E5 00     indtape.
2805 46 6F 75 6E 64 20 A0 00     Found66.
280D 4C 6F 61 64 69 6E E7 00     Loading.
2815 53 61 76 69 6E E7 00 00     Saving..
281D 4F EB 00 62 6C 6F 63 EB     Ok.block
2825 00 55 6E 6E 61 6D 65 E4     .Unnamed
282D 66 69 6C 65 20 20 20 A0     file
2835 00
```

*****CAS READ

```
2836 CD73p8      call 2873      allumer moteur ouvr. clavier
2839 F5          push af
283A 21B828      ld  hl,28B8
283D 1819        jr   2858
```

*****CAS WRITE

```
283F CD7328      call 2873      allumer moteur ouvr. clavier
2842 F5          push af
2843 CD6429      call 2964
2846 21F728      ld  hl,28F7
2849 DC9D28      call c,289D
284C DC7929      call c,2979
284F 180F        jr   2860
```

*****CAS CHECK

```
2851 CD7328      call 2873      allumer moteur ouvr. clavier
2854 F5          push af
2855 21C728      ld  hl,28C7
```

CASSETTE MANAGER

```

2858 E5          push  hl
2859 CD1929      call  2919
285C E1          pop   hl
285D DC9D28      call  c,289D
2860 D1          pop   de
2861 F5          push  af
2862 0182F7      ld    bc,F782      Port A=Out
2865 ED49       out   (c),c
2867 0110F6      ld    bc,F610      allumer moteur
286A ED49       out   (c),c
286C FB         ei
286D 7A         ld    a,d
286E CD512A     call  2A51          CAS RESTORE MOTOR
2871 F1         pop   af
2872 C9         ret

*****allumer moteur ouvr. clavier
2873 32CDB8      ld    (B8CD),a
2876 1B         dec   de
2877 1C         inc   e
2878 E5          push  hl
2879 D5          push  de
287A CD681E     call  1E68          SOUND RESET
287D D1         pop   de
287E DDE1       pop   ix
2880 CD4B2A     call  2A4B          CAS START MOTOR
2883 F3         di
2884 010EF4      ld    bc,F40E      Sound I/O Port select
2887 ED49       out   (c),c
2889 01D0F6      ld    bc,F6D0      Strobe mis
288C ED49       out   (c),c
288E 0E10       ld    c,10         Strobe coupé
2890 ED49       out   (c),c
2892 0192F7      ld    bc,F792      Port A=In
2895 ED49       out   (c),c
2897 0158F6      ld    bc,F658      ouvrir clavier Y9 (ESC)
289A ED49       out   (c),c      & sound I/O sur port A
289C C9         ret

289D 7A         ld    a,d

```

CASSETTE MANAGER

```

289E B7          or    a
289F 280D       Jr    z,28AE
28A1 E5         push hl
28A2 D5         push de
28A3 1E00       ld    e,00
28A5 CDAE28     call 28AE
28A8 D1         pop   de
28A9 E1         pop   hl
28AA D0         ret   nc
28AB 15         dec   d
28AC 20F3       Jr    nz,28A1
28AE 01FFFF     ld    bc,FFFF
28B1 ED43D3B8   ld    (B8D3),bc
28B5 1601       ld    d,01
28B7 E9         Jp    (hl)

28B8 CDB029     call 29B0
28BB D0         ret   nc
28BC DD7700     ld    (ix+00),a
28BF DD23       inc   ix
28C1 15         dec   d
28C2 1D         dec   e
28C3 20F3       Jr    nz,28B8
28C5 1812       Jr    28D9
28C7 CDB029     call 29B0
28CA D0         ret   nc
28CB 47         ld    b,a
28CC CDDCBA     call BADC (056D) RAM LAM (IX)
28CF A8         xor   b
28D0 3E03       ld    a,03
28D2 C0         ret   nz
28D3 DD23       inc   ix
28D5 15         dec   d
28D6 1D         dec   e
28D7 20EE       Jr    nz,28C7
28D9 15         dec   d
28DA 2806       Jr    z,28E2
28DC CDB029     call 29B0
28DF D0         ret   nc
28E0 18F7       Jr    28D9

```

CASSETTE MANAGER

28E2	CDA629	call	29A6
28E5	CDB029	call	29B0
28E8	D0	ret	nc
28E9	AA	xor	d
28EA	2007	jr	nz,28F3
28EC	CDB029	call	29B0
28EF	D0	ret	nc
28F0	AB	xor	e
28F1	37	scf	
28F2	C8	ret	z
28F3	3E02	ld	a,02
28F5	B7	or	a
28F6	C9	ret	
28F7	CDDCBA	call	BADC (056D) RAM LAM (IX)
28FA	CDF829	call	29F8
28FD	D0	ret	nc
28FE	DD23	inc	ix
2900	15	dec	d
2901	1D	dec	e
2902	20F3	jr	nz,28F7
2904	15	dec	d
2905	2807	jr	z,290E
2907	AF	xor	a
2908	CDF829	call	29F8
290B	D0	ret	nc
290C	18F6	jr	2904
290E	CDA629	call	29A6
2911	CDF829	call	29F8
2914	D0	ret	nc
2915	7B	ld	a,e
2916	C3F829	jp	29F8
2919	D5	push	de
291A	CD2329	call	2923
291D	D1	pop	de
291E	D8	ret	c
291F	B7	or	a
2920	C8	ret	z
2921	18F6	jr	2919

CASSETTE MANAGER

2923	2E55	ld	l,55	
2925	CDCD29	call	29CD	CAS Input RD DATA & Test ESC
2928	D0	ret	nc	
2929	110000	ld	de,0000	
292C	62	ld	h,d	
292D	CDCD29	call	29CD	CAS Input RD DATA & Test ESC
2930	D0	ret	nc	
2931	EB	ex	de,h1	
2932	0600	ld	b,00	
2934	09	add	h1,bc	
2935	EB	ex	de,h1	
2936	25	dec	h	
2937	20F4	jr	nz,292D	
2939	61	ld	h,c	
293A	79	ld	a,c	
293B	92	sub	d	
293C	4F	ld	c,a	
293D	9F	sbc	a,a	
293E	47	ld	b,a	
293F	EB	ex	de,h1	
2940	09	add	h1,bc	
2941	EB	ex	de,h1	
2942	CDCD29	call	29CD	CAS Input RD DATA & Test ESC
2945	D0	ret	nc	
2946	7A	ld	a,d	
2947	CB3F	sr1	a	
2949	CB3F	sr1	a	
294B	8A	adc	a,d	
294C	94	sub	h	
294D	38EA	jr	c,2939	
294F	91	sub	c	
2950	38E7	jr	c,2939	
2952	7A	ld	a,d	
2953	1F	rra		
2954	8A	adc	a,d	
2955	67	ld	h,a	
2956	22CEB8	ld	(B8CE),h1	
2959	CDB029	call	29B0	
295C	D0	ret	nc	
295D	21CDB8	ld	h1,B8CD	

CASSETTE MANAGER

2960	AE	xor	(hl)
2961	C0	ret	nz
2962	37	scf	
2963	C9	ret	
2964	CD892A	call	2A89
2967	210108	ld	hl,0801
296A	CD7C29	call	297C
296D	D0	ret	nc
296E	B7	or	a
296F	CD082A	call	2A08
2972	D0	ret	nc
2973	3ACDB8	ld	a,(B8CD)
2976	C3F829	jp	29F8
2979	212100	ld	hl,0021
297C	06F4	ld	b,F4
297E	ED78	in	a,(c)
2980	E604	and	04
2982	C8	ret	z
2983	E5	push	hl
2984	37	scf	
2985	CD082A	call	2A08
2988	E1	pop	hl
2989	2B	dec	hl
298A	7C	ld	a,h
298B	B5	or	l
298C	20EE	jr	nz,297C
298E	37	scf	
298F	C9	ret	
2990	2AD3B8	ld	hl,(B8D3)
2993	AC	xor	h
2994	F2A029	jp	p,29A0
2997	7C	ld	a,h
2998	EE08	xor	08
299A	67	ld	h,a
299B	7D	ld	a,l
299C	EE10	xor	10
299E	6F	ld	l,a

CASSETTE MANAGER

```

299F 37          scf
29A0 ED6A       adc   hl,hl
29A2 22D3B8     ld    (B8D3),hl
29A5 C9         ret

```

```

29A6 2AD3B8     ld    hl,(B8D3)
29A9 7D         ld    a,l
29AA 2F         cpl
29AB 5F         ld    e,a
29AC 7C         ld    a,h
29AD 2F         cpl
29AE 57         ld    d,a
29AF C9         ret
                -

```

```

29B0 D5         push  de
29B1 1E08       ld    e,08
29B3 2ACEB8     ld    hl,(B8CE)
29B6 CDD429     call  29D4
29B9 DCDD29     call  c,29DD
29BC 300D       jr    nc,29CB
29BE 7C         ld    a,h
29BF 91         sub   c
29C0 9F         sbc   a,a
29C1 CB12       rl    d
29C3 CD9029     call  2990
29C6 1D         dec   e
29C7 20EA       jr    nz,29B3
29C9 7A         ld    a,d
29CA 37         scf
29CB D1         pop  de
29CC C9         ret

```

*****CAS Input RD DATA & Test ESC

```

29CD 06F4       ld    b,F4      Port A
29CF ED78       in    a,(c)     Keyb X
29D1 E604       and   04        ESC ?
29D3 C8         ret    z        oui
29D4 ED5F       ld    a,r
29D6 C603       add   a,03
29D8 0F        rrca

```

CASSETTE MANAGER

29D9	0F	rrca	
29DA	E61F	and	1F
29DC	4F	ld	c,a
29DD	06F5	ld	b,F5 Port B
29DF	79	ld	a,c
29E0	C602	add	a,02
29E2	4F	ld	c,a
29E3	380E	jr	c,29F3
29E5	ED78	in	a,(c) Input RD DATA
29E7	AD	xor	1
29E8	E680	and	80
29EA	20F3	jr	nz,29DF
29EC	AF	xor	a
29ED	ED4F	ld	r,a
29EF	CB0D	rrc	1
29F1	37	scf	
29F2	C9	ret	
29F3	AF	xor	a
29F4	ED4F	ld	r,a
29F6	3C	inc	a
29F7	C9	ret	
29F8	D5	push	de
29F9	1E08	ld	e,08
29FB	57	ld	d,a
29FC	CB02	rlc	d
29FE	CD082A	call	2A08
2A01	3003	jr	nc,2A06
2A03	1D	dec	e
2A04	20F6	jr	nz,29FC
2A06	D1	pop	de
2A07	C9	ret	
2A08	ED4BDOB8	ld	bc,(B8D0)
2A0C	2AD2B8	ld	hl,(B8D2)
2A0F	9F	sbc	a,a
2A10	67	ld	h,a
2A11	2807	jr	z,2A1A
2A13	7D	ld	a,l

CASSETTE MANAGER

```

2A14 87          add  a,a
2A15 80          add  a,b
2A16 6F          ld   l,a
2A17 79          ld   a,c
2A18 90          sub  b
2A19 4F          ld   c,a
2A1A 7D          ld   a,l
2A1B 32D0B8     ld   (B8D0),a
2A1E 2E0A       ld   l,0A      WR DATA éteint
2A20 CD372A     call 2A37      CAS Output WR DATA
2A23 3806       jr   c,2A2B
2A25 91          sub  c
2A26 300C       jr   nc,2A34
2A28 2F         cpl
2A29 3C         inc  a
2A2A 4F         ld   c,a
2A2B 7C         ld   a,h
2A2C CD9029     call 2990
2A2F 2E0B       ld   l,0B      WR DATA mis
2A31 CD372A     call 2A37      CAS Output WR DATA
2A34 3E01       ld   a,01
2A36 C9         ret

```

*****CAS Output WR DATA

```

2A37 ED5F       ld   a,r
2A39 CB3F       srl  a
2A3B 91         sub  c
2A3C 3003       jr   nc,2A41
2A3E 3C         inc  a
2A3F 20FD       jr   nz,2A3E
2A41 06F7       ld   b,F7      Port Control
2A43 ED69       out  (c),l     WR DATA
2A45 F5         push af
2A46 AF         xor  a
2A47 ED4F       ld   r,a
2A49 F1         pop  af
2A4A C9         ret

```

*****CAS START MOTOR

```

2A4B 3E10       ld   a,10

```

CASSETTE MANAGER

```

2A4D 1802      Jr  2A51      CAS RESTORE MOTOR
*****CAS STOP MOTOR
2A4F 3EEF      ld  a,EF
*****CAS RESTORE MOTOR
2A51 C5        push bc
2A52 06F6      ld  b,F6      Port C
2A54 ED48      in  c,(c)
2A56 04        inc  b
2A57 E610      and  10
2A59 3E08      ld  a,08
2A5B 2801      Jr  z,2A5E
2A5D 3C        inc  a
2A5E ED79      out (c),a     Moteur allumé/éteint
2A60 37        scf
2A61 280C      Jr  z,2A6F
2A63 79        ld  a,c
2A64 E610      and  10
2A66 C5        push bc
2A67 01C800    ld  bc,00C8
2A6A 37        scf
2A6B CC722A    call z,2A72
2A6E C1        pop  bc
2A6F 79        ld  a,c
2A70 C1        pop  bc
2A71 C9        ret

2A72 C5        push bc
2A73 E5        push hl
2A74 CD892A    call 2A89
2A77 3E42      ld  a,42
2A79 CDBD1C    call 1CBD     KM TEST KEY
2A7C E1        pop  hl
2A7D C1        pop  bc
2A7E 2007      Jr  nz,2A87
2A80 0B        dec  bc
2A81 78        ld  a,b
2A82 B1        or   c
2A83 20ED      Jr  nz,2A72

```

CASSETTE MANAGER

2A85	37	scf	
2A86	C9	ret	
2A87	AF	xor	a
2A88	C9	ret	
2A89	018206	ld	bc,0682
2A8C	0B	dec	bc
2A8D	78	ld	a,b
2A8E	B1	or	c
2A8F	20FB	jr	nz,2A8C
2A91	C9	ret	
2A92	C7	rst	0
2A93	C7	rst	0
2A94	C7	rst	0
2A95	C7	rst	0
2A96	C7	rst	0
2A97	C7	rst	0

SCREEN EDITOR

2.5.10 SCREEN EDITOR (EDIT)

L'éditeur n'est pas en réalité un pack dans le sens où nous l'avons compris jusqu'ici. Il n'est en effet pas du tout utilisé par le système d'exploitation.

Il doit plutôt être considéré comme lié aux packs arithmétiques. De même que ceux-ci, l'éditeur n'est appelé que par le Basic.

Nous ne voyons pas quelles routines individuelles pourraient être utilisées, si ce n'est tout au plus l'éditeur lui-même globalement.

Il vous faut pour cela fournir à hl l'adresse de début du texte que vous souhaitez éditer. Ce texte doit comprendre un maximum de 255 caractères, ce qui correspond également à la taille maximum d'une ligne Basic.

SCREEN EDITOR

***** EDIT

```

2A98 C5      push  bc
2A99 D5      push  de
2A9A E5      push  hl
2A9B E5      push  hl      pointeur sur buffer d'entrée
2A9C 01FF00  ld    bc,00FF
2A9F 0C      inc   c        compteur caractères dans buffer
2AA0 7E      ld    a,(hl)
2AA1 23      inc   hl
2AA2 B7      or    a
2AA3 20FA   jr    nz,2A9F
2AA5 32DDB8  ld    (B8DD),a (Insert Flag)
2AA8 CD6F2C  call  2C6F
2AAB E1      pop   hl
2AAC CD672D  call  2D67
2AAF C5      push  bc
2AB0 E5      push  hl
2AB1 CDD92D  call  2DD9      caractère du clavier
2AB4 E1      pop   hl
2AB5 C1      pop   bc
2AB6 CDC62A  call  2AC6      exécuter saut EDIT
2AB9 30F4   jr    nc,2AAF
2ABB F5      push  af
2ABC CDD22C  call  2CD2
2ABF F1      pop   af
2AC0 E1      pop   hl
2AC1 D1      pop   de
2AC2 C1      pop   bc
2AC3 FEFC   cp    FC
2AC5 C9      ret

```

***** exécuter saut EDIT

```

2AC6 E5      push  hl
2AC7 21E02A  ld    hl,2AE0  EDIT Table de saut 1
2ACA 5F      ld    e,a
2ACB 78      ld    a,b
2ACC B1      or    c        caractère dans buffer?
2ACD 7B      ld    a,e
2ACE 200B   jr    nz,2ADB  oui =>
2AD0 FEFO   cp    FO      une des touches curseur?

```

SCREEN EDITOR

2AD2	3807	Jr	c,2ADB	non =>
2AD4	FEF4	cp	F4	touche curseur & SHFT/CTRL ?
2AD6	3003	Jr	nc,2ADB	oui =>
2AD8	211C2B	ld	hl,2B1C	EDIT Table de saut 2
2ADB	CDF62D	call	2DF6	aller chercher adr. de saut EDIT
2ADE	E3	ex	(sp),hl	manipuler adr. ret
2ADF	C9	ret		Saut d'après table

***** EDIT Table de saut 1

2AE0	13	db	13	Nombre d'entrées
2AE1	012C	dw	2C01	insérer caractère
2AE3	FC	db	FC	
2AE4	422B	dw	2B42	ESC
2AE6	EF	db	EF	
2AE7	402B	dw	2B40	aucun effet
2AE9	0D	db	0D	
2AEA	692B	dw	2B69	ENTER
2AEC	F0	db	F0	
2AED	B32B	dw	2BB3	CRSR UP (Buffer)
2AEF	F1	db	F1	
2AF0	7E2B	dw	2B7E	CRSR DWN (Buffer)
2AF2	F2	db	F2	
2AF3	AA2B	dw	2BAA	CRSR LEFT (Buffer)
2AF5	F3	db	F3	
2AF6	752B	dw	2B75	CRSR RGHT (Buffer)
2AF8	F8	db	F8	
2AF9	C72B	dw	2BC7	CTRL & CRSR UP
2AFB	F9	db	F9	
2AFC	922B	dw	2B92	CTRL & CRSR DWN
2AFE	FA	db	FA	
2AFF	BD2B	dw	2BBD	CTRL & CRSR LEFT
2B01	FB	db	FB	
2B02	892B	dw	2B89	CTRL & CRSR RGHT
2B04	F4	db	F4	
2B05	A22C	dw	2CA2	SHFT & CRSR UP
2B07	F5	db	F5	
2B08	A72C	dw	2CA7	SHFT & CRSR DWN
2B0A	F6	db	F6	
2B0B	9D2C	dw	2C9D	SHFT & CRSR LEFT
2B0D	F7	db	F7	

SCREEN EDITOR

2B0E	982C	dw	2C98	SHFT & CRSR RIGHT
2B10	E0	db	E0	
2B11	EA2C	dw	2CEA	COPY
2B13	7F	db	7F	
2B14	3D2C	dw	2C3D	DEL
2B16	10	db	10	
2B17	4A2C	dw	2C4A	CLR
2B19	E1	db	E1	
2B1A	F92B	dw	2BF9	CTRL & TAB (Flip Insert)

***** EDIT Table de saut 2

2B1C	04	db	04	Nombre d'entrées
2B1D	2B2B	-dw	2B2B	KLINGEL
2B1F	F0	db	F0	
2B20	2F2B	dw	2B2F	CRSR UP
2B22	F1	db	F1	
2B23	332B	dw	2B33	CRSR DWN
2B25	F2	db	F2	
2B26	3B2B	dw	2B3B	CRSR LEFT
2B28	F3	db	F3	
2B29	372B	dw	2B37	CRSR RIGHT

***** BIP

2B2B	3E07	ld	a,07	BEL
2B2D	180E	jr	2B3D	

***** CRSR UP

2B2F	3E0B	ld	a,0B	
2B31	180A	jr	2B3D	

***** CRSR DWN

2B33	3E0A	ld	a,0A	
2B35	1806	jr	2B3D	

***** CRSR RIGHT

2B37	3E09	ld	a,09	
2B39	1802	jr	2B3D	

***** CRSR LEFT

2B3B	3E08	ld	a,08	
------	------	----	------	--

SCREEN EDITOR

```
2B3D CD0014      call 1400      TXT OUTPUT
2B40 B7          or   a
2B41 C9          ret
```

***** ESC

```
2B42 F5          push af
2B43 CD492B      call 2B49
2B46 F1          pop  af
2B47 37          scf
2B48 C9          ret

2B49 CD692B      call 2B69      ENTER
2B4C 21612B      ld   hl,2B61  message *BREAK*
2B4F CD692B      call 2B69      ENTER
2B52 CD8011      call 1180      TXT GET CURSOR
2B55 25          dec  h
2B56 C8          ret  z
2B57 3E0D        ld   a,0D      CR
2B59 CD0014      call 1400      TXT OUTPUT
2B5C 3E0A        ld   a,0A      LF
2B5E C30014      jp   1400      TXT OUTPUT
```

***** message *BREAK*

```
2B61 2A 42 72 65 61 6B 2A 00      *Break*.
```

***** ENTER

```
2B69 F5          push af
2B6A 7E          ld   a,(hl)
2B6B 23          inc  hl
2B6C B7          or   a
2B6D C4A82D      call nz,2DA8
2B70 20F8        jr   nz,2B6A
2B72 F1          pop  af
2B73 37          scf
2B74 C9          ret
```

***** CRSR RGHT (Buffer)

```
2B75 1601        ld   d,01
```

SCREEN EDITOR

```
2B77 CD932B      call 2B93
2B7A CA2B2B      jp   z,2B2B     BIP
2B7D C9          ret
```

***** CRSR DWN (Buffer)

```
2B7E CDEB2B      call 2BEB
2B81 79          ld   a,c
2B82 90          sub  b
2B83 BA          cp   d
2B84 DA2B2B      jp   c,2B2B     BIP
2B87 180A        jr   2B93
```

***** CTRL & CRSR RGHT

```
2B89 CDEB2B      call 2BEB
2B8C 7A          ld   a,d
2B8D 93          sub  e
2B8E C8          ret  z
2B8F 57          ld   d,a
2B90 1801        jr   2B93
```

***** CTRL & CRSR DWN

```
2B92 51          ld   d,c
2B93 78          ld   a,b
2B94 B9          cp   c
2B95 C8          ret  z
2B96 D5          push de
2B97 CD502D      call 2D50
2B9A 7E          ld   a,(hl)
2B9B D4A82D      call nc,2DA8
2B9E 04          inc  b
2B9F 23          inc  hl
2BA0 D4672D      call nc,2D67
2BA3 D1          pop  de
2BA4 15          dec  d
2BA5 20EC        jr   nz,2B93
2BA7 F6FF        or   FF
2BA9 C9          ret
```

***** CRSR LEFT (Buffer)

```
2BAA 1601        ld   d,01
```

SCREEN EDITOR

```

2BAC CDC82B      call 2BC8
2BAF CA2B2B      jp   z,2B2B      BIP
2BB2 C9          ret

```

***** CTRL & CRSR UP (Buffer)

```

2BB3 CDEB2B      call 2BEB
2BB6 78          ld   a,b
2BB7 BA          cp   d
2BB8 DA2B2B      jp   c,2B2B      BIP
2BBB 180B        jr   2BC8

```

***** CTRL & CRSR LEFT

```

2BBD CDEB2B      call 2BEB
2BC0 7B          ld   a,e
2BC1 D601        sub  01
2BC3 C8          ret   z
2BC4 57          ld   d,a
2BC5 1801        jr   2BC8

```

***** CTRL & CRSR UP

```

2BC7 51          ld   d,c
2BC8 78          ld   a,b
2BC9 B7          or   a
2BCA C8          ret   z
2BCB CD4A2D      call 2D4A
2BCE 3007        jr   nc,2BD7
2BD0 05          dec  b
2BD1 2B          dec  hl
2BD2 15          dec  d
2BD3 20F3        jr   nz,2BC8
2BD5 1811        jr   2BE8
2BD7 78          ld   a,b
2BD8 B7          or   a
2BD9 280A        jr   z,2BE5
2BDB 05          dec  b
2BDC 2B          dec  hl
2BDD D5          push de
2BDE CD292D      call 2D29
2BE1 D1          pop  de
2BE2 15          dec  d

```

SCREEN EDITOR

```

2BE3 20F2      Jr   nz,2BD7
2BE5 CD672D    call 2D67
2BE8 F6FF      or   FF
2BEA C9        ret

2BEB E5         push hl
2BEC CD5612    call 1256      TXT GET WINDOW
2BEF 7A        ld   a,d
2BF0 94        sub  h
2BF1 3C        inc  a
2BF2 57        ld   d,a
2BF3 CD8011    call 1180      TXT GET CURSOR
2BF6 5C        ld   e,h
2BF7 E1        pop  hl
2BF8 C9        ret

```

***** CTRL & TAB (Flip Insert)

```

2BF9 3ADD8      ld   a,(B8DD) (Insert Flag)
2BFC 2F        cpl
2BFD 32DDB8     ld   (B8DD),a (Insert Flag)
2C00 C9        ret

```

***** insérer caractère

```

2C01 B7         or   a
2C02 C8        ret  z
2C03 5F        ld   e,a
2C04 3ADD8      ld   a,(B8DD) (Insert Flag)
2C07 B7         or   a
2C08 280D      Jr   z,2C17
2COA 78        ld   a,b
2COB B9        cp   c
2C0C 2809      Jr   z,2C17
2COE 73        ld   (hl),e
2COF 7B        ld   a,e
2C10 CDA82D    call 2DA8
2C13 23        inc  hl
2C14 04        inc  b
2C15 B7         or   a
2C16 C9        ret

```

SCREEN EDITOR

```

2C17 79          ld  a,c
2C18 FEFF       cp  FF
2C1A CA2B2B     jp  z,2B2B    BIP
2C1D AF         xor  a
2C1E 32DCB8     ld  (B8DC),a
2C21 7B         ld  a,e
2C22 CDA82D     call 2DA8
2C25 0C         inc  c
2C26 E5         push hl
2C27 7E         ld  a,(hl)
2C28 73         ld  (hl),e
2C29 5F         ld  e,a
2C2A 23         inc  hl
2C2B B7         or   a
2C2C 20F9       jr  nz,2C27
2C2E 77         ld  (hl),a
2C2F E1         pop  hl
2C30 04         inc  b
2C31 23         inc  hl
2C32 CD672D     call 2D67
2C35 3ADC88     ld  a,(B8DC)
2C38 B7         or   a
2C39 C4292D     call nz,2D29
2C3C C9         ret

```

***** DEL

```

2C3D 78          ld  a,b
2C3E B7         or   a
2C3F CA2B2B     jp  z,2B2B    BIP
2C42 CD4A2D     call 2D4A
2C45 D22B2B     jp  nc,2B2B   BIP
2C48 05         dec  b
2C49 2B         dec  hl

```

***** CLR

```

2C4A 78          ld  a,b
2C4B B9         cp  c
2C4C CA2B2B     jp  z,2B2B    BIP
2C4F E5         push hl
2C50 23         inc  hl

```


SCREEN EDITOR

2C51	7E	ld	a,(h1)
2C52	2B	dec	h1
2C53	77	ld	(h1),a
2C54	23	inc	h1
2C55	B7	or	a
2C56	20F8	Jr	nz,2C50
2C58	2B	dec	h1
2C59	3620	ld	(h1),20
2C5B	32DCB8	ld	(B8DC),a
2C5E	E3	ex	(sp),h1
2C5F	CD672D	call	2D67
2C62	E3	ex	(sp),h1
2C63	3600	ld	(h1),00
2C65	E1	pop	h1
2C66	0D	dec	c
2C67	3ADC8B	ld	a,(B8DC)
2C6A	B7	or	a
2C6B	C42D2D	call	nz,2D2D
2C6E	C9	ret	
2C6F	210000	ld	h1,0000
2C72	22DEB8	ld	(B8DE),h1
2C75	C9	ret	
2C76	ED5BDEB8	ld	de,(B8DE)
2C7A	7C	ld	a,h
2C7B	AA	xor	d
2C7C	C0	ret	nz
2C7D	7D	ld	a,l
2C7E	AB	xor	e
2C7F	C0	ret	nz
2C80	37	scf	
2C81	C9	ret	
2C82	4F	ld	c,a
2C83	2ADEB8	ld	h1,(B8DE)
2C86	7C	ld	a,h
2C87	B5	or	l
2C88	C8	ret	z
2C89	7D	ld	a,l

SCREEN EDITOR

```

2C8A 81          add  a,c
2C8B 6F          ld   l,a
2C8C CDCE11     call 11CE    TXT VALIDATE
2C8F 3803       jr   c,2C94
2C91 210000     ld   hl,0000
2C94 22DEB8     ld   (B8DE),hl
2C97 C9         ret

***** SHFT & CRSR RIGHT
2C98 110001     ld   de,0100
2C9B 180D       jr   2CAA

***** SHFT & CRSR LEFT
2C9D 1100FF     ld   de,FF00
2CA0 1808       jr   2CAA

***** SHFT & CRSR UP
2CA2 11FF00     ld   de,00FF
2CA5 1803       jr   2CAA

***** SHFT & CRSR DWN
2CA7 110100     ld   de,0001
2CAA C5         push bc
2CAB E5         push hl
2CAC 2ADEB8     ld   hl,(B8DE)
2CAF 7C         ld   a,h
2CB0 B5         or   l
2CB1 CC8011     call z,1180    TXT GET CURSOR
2CB4 7C         ld   a,h
2CB5 82         add  a,d
2CB6 67         ld   h,a
2CB7 7D         ld   a,l
2CB8 83         add  a,e
2CB9 6F         ld   l,a
2CBA CDCE11     call 11CE    TXT VALIDATE
2CBD 300B       jr   nc,2CCA
2CBF E5         push hl
2CC0 CDD22C     call 2CD2
2CC3 E1         pop  hl
2CC4 22DEB8     ld   (B8DE),hl

```

SCREEN EDITOR

```

2CC7 CDCD2C      call 2CCD
2CCA E1          pop  hl
2CCB C1          pop  bc
2CCC C9          ret

2CCD 116812      ld   de,1268   TXT PLACE/REMOVE CURSOR
2CD0 1803        jr   2CD5
2CD2 116812      ld   de,1268   TXT PLACE/REMOVE CURSOR
2CD5 2ADEB8      ld   hl,(B8DE)
2CD8 7C          ld   a,h
2CD9 B5          or   l
2CDA C8          ret   z
2CDB E5          push hl
2CDC CD8011      call 1180      TXT GET CURSOR
2CDF E3          ex   (sp),hl
2CE0 CD7411      call 1174      TXT SET CURSOR
2CE3 CD1600      call 0016
2CE6 E1          pop  hl
2CE7 C37411      jp   1174      TXT SET CURSOR

```

***** COPY

```

2CEA C5          push bc
2CEB E5          push hl
2CEC CD8011      call 1180      TXT GET CURSOR
2CEF EB          ex   de,hl
2CF0 2ADEB8      ld   hl,(B8DE)
2CF3 7C          ld   a,h
2CF4 B5          or   l
2CF5 200C        jr   nz,2D03
2CF7 78          ld   a,b
2CF8 B1          or   c
2CF9 2026        jr   nz,2D21
2CFB CD8011      call 1180      TXT GET CURSOR
2CFE 22DEB8      ld   (B8DE),hl
2D01 1806        jr   2D09
2D03 CD7411      call 1174      TXT SET CURSOR
2D06 CD6812      call 1268      TXT PLACE/REMOVE CURSOR
2D09 CDAB13      call 13AB      TXT RD CHAR
2DOC F5          push af
2DOD EB          ex   de,hl

```

SCREEN EDITOR

2D0E	CD7411	call	1174	TXT SET CURSOR
2D11	2ADEB8	ld	hl, (B8DE)	
2D14	24	inc	h	
2D15	CDCE11	call	11CE	TXT VALIDATE
2D18	3003	Jr	nc, 2D1D	
2D1A	22DEB8	ld	(B8DE), hl	
2D1D	CDCD2C	call	2CCD	
2D20	F1	pop	af	
2D21	E1	pop	hl	
2D22	C1	pop	bc	
2D23	DA012C	jp	c, 2C01	insérer caractère
2D26	C32B2B	jp	2B2B	BIP
2D29	1601	ld	d, 01	
2D2B	1802	Jr	2D2F	
2D2D	16FF	ld	d, FF	
2D2F	C5	push	bc	
2D30	E5	push	hl	
2D31	D5	push	de	
2D32	CDD22C	call	2CD2	
2D35	D1	pop	de	
2D36	2ADEB8	ld	hl, (B8DE)	
2D39	7C	ld	a, h	
2D3A	B5	or	l	
2D3B	2809	Jr	z, 2D46	
2D3D	7C	ld	a, h	
2D3E	82	add	a, d	
2D3F	67	ld	h, a	
2D40	CD8C2C	call	2C8C	
2D43	CDCD2C	call	2CCD	
2D46	E1	pop	hl	
2D47	C1	pop	bc	
2D48	B7	or	a	
2D49	C9	ret		
2D4A	D5	push	de	
2D4B	1108FF	ld	de, FF08	
2D4E	1804	Jr	2D54	
2D50	D5	push	de	
2D51	110901	ld	de, 0109	
2D54	C5	push	bc	

SCREEN EDITOR

```

2D55 E5      push  hl
2D56 CD8011  call  1180      TXT GET CURSOR
2D59 7A      ld    a,d
2D5A 84      add   a,h
2D5B 67      ld    h,a
2D5C CDCE11  call  11CE      TXT VALIDATE
2D5F 7B      ld    a,e
2D60 DC0014  call  c,1400    TXT OUTPUT
2D63 E1      pop   hl
2D64 C1      pop   bc
2D65 D1      pop   de
2D66 C9      ret

2D67 C5      push  bc
2D68 E5      push  hl
2D69 EB      ex    de,hl
2D6A CD8011  call  1180      TXT GET CURSOR
2D6D 4F      ld    c,a
2D6E EB      ex    de,hl
2D6F 7E      ld    a,(hl)
2D70 23      inc   hl
2D71 B7      or    a
2D72 C4852D  call  nz,2D85
2D75 20F8     jr    nz,2D6F
2D77 CD8011  call  1180      TXT GET CURSOR
2D7A 91      sub   c
2D7B EB      ex    de,hl
2D7C 85      add   a,l
2D7D 6F      ld    l,a
2D7E CD7411  call  1174      TXT SET CURSOR
2D81 E1      pop   hl
2D82 C1      pop   bc
2D83 B7      or    a
2D84 C9      ret

2D85 F5      push  af
2D86 C5      push  bc
2D87 D5      push  de
2D88 E5      push  hl
2D89 47      ld    b,a

```

SCREEN EDITOR

2D8A	CD8011	call	1180	TXT GET CURSOR
2D8D	91	sub	c	
2D8E	83	add	a,e	
2D8F	5F	ld	e,a	
2D90	48	ld	c,b	
2D91	CDCE11	call	11CE	TXT VALIDATE
2D94	3805	Jr	c,2D9B	
2D96	78	ld	a,b	
2D97	87	add	a,a	
2D98	3C	inc	a	
2D99	83	add	a,e	
2D9A	5F	ld	e,a	
2D9B	EB	ex	de,h1	
2D9C	CDCE11	call	11CE	TXT VALIDATE
2D9F	79	ld	a,c	
2DA0	DCA82D	call	c,2DA8	
2DA3	E1	pop	h1	
2DA4	D1	pop	de	
2DA5	C1	pop	bc	
2DA6	F1	pop	af	
2DA7	C9	ret		
2DA8	F5	push	af	
2DA9	C5	push	bc	
2DAA	D5	push	de	
2DAB	E5	push	h1	
2DAC	47	ld	b,a	
2DAD	CD8011	call	1180	TXT GET CURSOR
2DB0	4F	ld	c,a	
2DB1	C5	push	bc	
2DB2	CDCE11	call	11CE	TXT VALIDATE
2DB5	C1	pop	bc	
2DB6	DC762C	call	c,2C76	
2DB9	F5	push	af	
2DBA	DCD22C	call	c,2CD2	
2DBD	78	ld	a,b	
2DBE	C5	push	bc	
2DBF	CD3413	call	1334	TXT WR CHAR
2DC2	C1	pop	bc	
2DC3	CD8011	call	1180	TXT GET CURSOR

SCREEN EDITOR

```

2DC6 91          sub  c
2DC7 C4822C     call nz,2C82
2DCA F1         pop  af
2DCB 3007       Jr   nc,2DD4
2DCD 9F         sbc  a,a
2DCE 32DCB8     ld   (B8DC),a
2DD1 CDCD2C     call 2CCD
2DD4 E1         pop  hl
2DD5 D1         pop  de
2DD6 C1         pop  bc
2DD7 F1         pop  af
2DD8 C9         ret

```

***** caractère du clavier

```

2DD9 CD8011     call 1180    TXT GET CURSOR
2DDC 4F         ld   c,a
2DDD CDCE11     call 11CE    TXT VALIDATE
2DE0 CD762C     call 2C76
2DE3 DA3C1A     Jp   c,1A3C  KM WAIT CHAR
2DE6 CD7912     call 1279    TXT CUR ON
2DE9 CD8011     call 1180    TXT GET CURSOR
2DEC 91         sub  c
2DED C4822C     call nz,2C82
2DF0 CD3C1A     call 1A3C    KM WAIT CHAR
2DF3 C38112     Jp   1281    TXT CUR OFF

```

***** aller chercher adr. saut EDIT

```

2DF6 F5         push af
2DF7 C5         push bc
2DF8 46         ld   b,(hl)
2DF9 23         inc  hl
2DFA E5         push hl
2DFB 23         inc  hl
2DFC 23         inc  hl
2DFD BE         cp   (hl)
2DFE 23         inc  hl
2DFF 2804       Jr   z,2E05
2E01 05         dec  b
2E02 20F7       Jr   nz,2DFB
2E04 E3         ex   (sp),hl

```

SCREEN EDITOR

2E05	F1	pop	af
2E06	7E	ld	a,(h1)
2E07	23	inc	h1
2E08	66	ld	h,(h1)
2E09	6F	ld	l,a
2E0A	C1	pop	bc
2E0B	F1	pop	af
2E0C	C9	ret	
2E0D	C7	rst	0
2E0E	C7	rst	0
2E0F	C7	rst	0
2E10	C7	rst	0
2E11	C7	rst	0
2E12	C7	rst	0
2E13	C7	rst	0
2E14	C7	rst	0
2E15	C7	rst	0
2E16	C7	rst	0
2E17	C7	rst	0

CHARACTERS

2.6 Le générateur de caractères

Ce n'est pas que nous voulions à tout prix abuser de votre patience avec les pages suivantes ni que nous pensions que l'ouvrage ne comporte pas encore assez de pages.

Nous pensons simplement que le jeu de caractères est un outil de travail important auquel s'appliquent même spécialement certaines instructions du jeu d'instructions Basic.

Pour que vous n'ayez pas à réinventer la poudre chaque fois que vous utilisez ces instructions, par exemple lorsque vous voulez produire des accents, il vous suffit de rechercher la forme du 'e' et de rajouter au dessus les points qui formeront l'accent aigu ou grave. Il vous suffit alors d'utiliser les valeurs ainsi calculées dans votre instruction de définition d'un caractère.

Nous nous permettons de vous donner un petit conseil. Vous constaterez que la plupart des dessins figurant dans les pages suivantes marquent toujours les lignes verticales par une paire de points (deux points sur la même ligne horizontale). Il vaut mieux éviter en effet de constituer des lignes verticales n'ayant qu'un point de largeur. En effet un point isolé est difficile à discerner à l'écran, surtout si vous disposez d'un moniteur couleur.

Mais maintenant vous pouvez donner libre cours à votre imagination et faire vos propres expériences.
















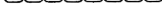














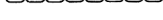















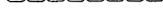

















































CHARACTERS

3800	FF		3808	FF	
3801	C3		3809	C0	
3802	C3		380A	C0	
3803	C3		380B	C0	
3804	C3		380C	C0	
3805	C3		380D	C0	
3806	C3		380E	C0	
3807	FF		380F	C0	
3810	18		3818	03	
3811	18		3819	03	
3812	18		381A	03	
3813	18		381B	03	
3814	18		381C	03	
3815	18		381D	03	
3816	18		381E	03	
3817	FF		381F	FF	
3820	0C		3828	FF	
3821	18		3829	C3	
3822	30		382A	E7	
3823	7E		382B	DB	
3824	0C		382C	DB	
3825	18		382D	E7	
3826	30		382E	C3	
3827	00		382F	FF	
3830	00		3838	3C	
3831	01		3839	66	
3832	03		383A	C3	
3833	06		383B	C3	
3834	CC		383C	FF	
3835	78		383D	24	
3836	30		383E	E7	
3837	00		383F	00	
3840	00		3848	00	
3841	00		3849	00	
3842	30		384A	0C	
3843	60		384B	06	
3844	FF		384C	FF	
3845	60		384D	06	
3846	30		384E	0C	
3847	00		384F	00	
3850	18		3858	18	
3851	18		3859	3C	
3852	18		385A	7E	
3853	18		385B	DB	
3854	DB		385C	18	
3855	7E		385D	18	
3856	3C		385E	18	
3857	18		385F	18	















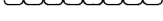










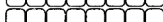










































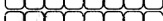















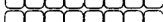











CHARACTERS

3860	18		3868	00	
3861	5A		3869	03	
3862	3C		386A	33	
3863	99		386B	63	
3864	DB		386C	FE	
3865	7E		386D	60	
3866	3C		386E	30	
3867	18		386F	00	
3870	3C		3878	3C	
3871	66		3879	66	
3872	FF		387A	C3	
3873	DB		387B	DB	
3874	DB		387C	DB	
3875	FF		387D	C3	
3876	66		387E	66	
3877	3C		387F	3C	
3880	FF		3888	3C	
3881	C3		3889	7E	
3882	C3		388A	DB	
3883	FF		388B	DB	
3884	C3		388C	DF	
3885	C3		388D	C3	
3886	C3		388E	66	
3887	FF		388F	3C	
3890	3C		3898	3C	
3891	66		3899	66	
3892	C3		389A	C3	
3893	DF		389B	FB	
3894	DB		389C	DB	
3895	DB		389D	DB	
3896	7E		389E	7E	
3897	3C		389F	3C	
38A0	3C		38A8	00	
38A1	7E		38A9	01	
38A2	DB		38AA	33	
38A3	DB		38AB	1E	
38A4	FB		38AC	CE	
38A5	C3		38AD	7B	
38A6	66		38AE	31	
38A7	3C		38AF	00	
38B0	7E		38B8	03	
38B1	66		38B9	03	
38B2	66		38BA	03	
38B3	66		38BB	FF	
38B4	66		38BC	03	
38B5	66		38BD	03	
38B6	66		38BE	03	
38B7	E7		38BF	00	















































































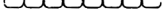















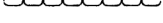

CHARACTERS

38C0	FF		38C8	18	
38C1	66		38C9	18	
38C2	3C		38CA	3C	
38C3	18		38CB	3C	
38C4	18		38CC	3C	
38C5	3C		38CD	3C	
38C6	66		38CE	18	
38C7	FF		38CF	18	
38D0	3C		38D8	3C	
38D1	66		38D9	66	
38D2	66		38DA	C3	
38D3	30		38DB	FF	
38D4	18		38DC	C3	
38D5	00		38DD	C3	
38D6	18		38DE	66	
38D7	00		38DF	3C	
38E0	FF		38E8	FF	
38E1	DB		38E9	C3	
38E2	DB		38EA	C3	
38E3	DB		38EB	FB	
38E4	FB		38EC	DB	
38E5	C3		38ED	DB	
38E6	C3		38EE	DB	
38E7	FF		38EF	FF	
38F0	FF		38F8	FF	
38F1	C3		38F9	DB	
38F2	C3		38FA	DB	
38F3	DF		38FB	DB	
38F4	DB		38FC	DF	
38F5	DB		38FD	C3	
38F6	DB		38FE	C3	
38F7	FF		38FF	FF	
3900	00		3908	18	
3901	00		3909	18	
3902	00		390A	18	
3903	00		390B	18	
3904	00		390C	18	
3905	00		390D	00	
3906	00		390E	18	
3907	00		390F	00	
3910	6C		3918	6C	
3911	6C		3919	6C	
3912	6C		391A	FE	
3913	00		391B	6C	
3914	00		391C	FE	
3915	00		391D	6C	
3916	00		391E	6C	
3917	00		391F	00	

CHARACTERS

3920	18		3928	00	
3921	3E		3929	C6	
3922	58		392A	CC	
3923	3C		392B	18	
3924	1A		392C	30	
3925	7C		392D	66	
3926	18		392E	C6	
3927	00		392F	00	
3930	38		3938	18	
3931	6C		3939	18	
3932	38		393A	30	
3933	76		393B	00	
3934	DC		393C	00	
3935	CC		393D	00	
3936	76		393E	00	
3937	00		393F	00	
3940	0C		3948	30	
3941	18		3949	18	
3942	30		394A	0C	
3943	30		394B	0C	
3944	30		394C	0C	
3945	18		394D	18	
3946	0C		394E	30	
3947	00		394F	00	
3950	00		3958	00	
3951	66		3959	18	
3952	3C		395A	18	
3953	FF		395B	7E	
3954	3C		395C	18	
3955	66		395D	18	
3956	00		395E	00	
3957	00		395F	00	
3960	00		3968	00	
3961	00		3969	00	
3962	00		396A	00	
3963	00		396B	7E	
3964	00		396C	00	
3965	18		396D	00	
3966	18		396E	00	
3967	30		396F	00	
3970	00		3978	06	
3971	00		3979	0C	
3972	00		397A	18	
3973	00		397B	30	
3974	00		397C	60	
3975	18		397D	C0	
3976	18		397E	80	
3977	00		397F	00	

CHARACTERS

3980	7C		3988	18	
3981	C6		3989	38	
3982	CE		398A	18	
3983	D6		398B	18	
3984	E6		398C	18	
3985	C6		398D	18	
3986	7C		398E	7E	
3987	00		398F	00	
3990	3C		3998	3C	
3991	66		3999	66	
3992	06		399A	06	
3993	3C		399B	1C	
3994	60		399C	06	
3995	66		399D	66	
3996	7E		399E	3C	
3997	00		399F	00	
39A0	1C		39A8	7E	
39A1	3C		39A9	62	
39A2	6C		39AA	60	
39A3	CC		39AB	7C	
39A4	FE		39AC	06	
39A5	0C		39AD	66	
39A6	1E		39AE	3C	
39A7	00		39AF	00	
39B0	3C		39B8	7E	
39B1	66		39B9	66	
39B2	60		39BA	06	
39B3	7C		39BB	0C	
39B4	66		39BC	18	
39B5	66		39BD	18	
39B6	3C		39BE	18	
39B7	00		39BF	00	
39C0	3C		39C8	3C	
39C1	66		39C9	66	
39C2	66		39CA	66	
39C3	3C		39CB	3E	
39C4	66		39CC	06	
39C5	66		39CD	66	
39C6	3C		39CE	3C	
39C7	00		39CF	00	
39D0	00		39D8	00	
39D1	00		39D9	00	
39D2	18		39DA	18	
39D3	18		39DB	18	
39D4	00		39DC	00	
39D5	18		39DD	18	
39D6	18		39DE	18	
39D7	00		39DF	30	

CHARACTERS

39E0	0C		39E8	00	
39E1	18		39E9	00	
39E2	30		39EA	7E	
39E3	60		39EB	00	
39E4	30		39EC	00	
39E5	18		39ED	7E	
39E6	0C		39EE	00	
39E7	00		39EF	00	
39F0	60		39F8	3C	
39F1	30		39F9	66	
39F2	18		39FA	66	
39F3	0C		39FB	0C	
39F4	18		39FC	18	
39F5	30		39FD	00	
39F6	60		39FE	18	
39F7	00		39FF	00	
3A00	7C		3A08	18	
3A01	C6		3A09	3C	
3A02	DE		3A0A	66	
3A03	DE		3A0B	66	
3A04	DE		3A0C	7E	
3A05	C0		3A0D	66	
3A06	7C		3A0E	66	
3A07	00		3A0F	00	
3A10	FC		3A18	3C	
3A11	66		3A19	66	
3A12	66		3A1A	C0	
3A13	7C		3A1B	C0	
3A14	66		3A1C	C0	
3A15	66		3A1D	66	
3A16	FC		3A1E	3C	
3A17	00		3A1F	00	
3A20	F8		3A28	FE	
3A21	6C		3A29	62	
3A22	66		3A2A	68	
3A23	66		3A2B	78	
3A24	66		3A2C	68	
3A25	6C		3A2D	62	
3A26	F8		3A2E	FE	
3A27	00		3A2F	00	
3A30	FE		3A38	3C	
3A31	62		3A39	66	
3A32	68		3A3A	C0	
3A33	78		3A3B	C0	
3A34	68		3A3C	CE	
3A35	60		3A3D	66	
3A36	F0		3A3E	3E	
3A37	00		3A3F	00	

CHARACTERS

3A40	66		3A48	7E	
3A41	66		3A49	18	
3A42	66		3A4A	18	
3A43	7E		3A4B	18	
3A44	66		3A4C	18	
3A45	66		3A4D	18	
3A46	66		3A4E	7E	
3A47	00		3A4F	00	
3A50	1E		3A58	E6	
3A51	0C		3A59	66	
3A52	0C		3A5A	6C	
3A53	0C		3A5B	78	
3A54	CC		3A5C	6C	
3A55	CC		3A5D	66	
3A56	78		3A5E	E6	
3A57	00		3A5F	00	
3A60	F0		3A68	C6	
3A61	60		3A69	EE	
3A62	60		3A6A	FE	
3A63	60		3A6B	FE	
3A64	62		3A6C	D6	
3A65	66		3A6D	C6	
3A66	FE		3A6E	C6	
3A67	00		3A6F	00	
3A70	C6		3A78	38	
3A71	E6		3A79	6C	
3A72	F6		3A7A	C6	
3A73	DE		3A7B	C6	
3A74	CE		3A7C	C6	
3A75	C6		3A7D	6C	
3A76	C6		3A7E	38	
3A77	00		3A7F	00	
3A80	FC		3A88	38	
3A81	66		3A89	6C	
3A82	66		3A8A	C6	
3A83	7C		3A8B	C6	
3A84	60		3A8C	DA	

CHARACTERS

3AA0	7E		3AA8	66	
3AA1	5A		3AA9	66	
3AA2	18		3AAA	66	
3AA3	18		3AAB	66	
3AA4	18		3AAC	66	
3AA5	18		3AAD	66	
3AA6	3C		3AAE	3C	
3AA7	00		3AAF	00	
3AB0	66		3AB8	C6	
3AB1	66		3AB9	C6	
3AB2	66		3ABA	C6	
3AB3	66		3ABB	D6	
3AB4	66		3ABC	FE	
3AB5	3C		3ABD	EE	
3AB6	18		3ABE	C6	
3AB7	00		3ABF	00	
3AC0	C6		3AC8	66	
3AC1	6C		3AC9	66	
3AC2	38		3ACA	66	
3AC3	38		3ACB	3C	
3AC4	6C		3ACC	18	
3AC5	C6		3ACD	18	
3AC6	C6		3ACE	3C	
3AC7	00		3ACF	00	
3AD0	FE		3AD8	3C	
3AD1	C6		3AD9	30	
3AD2	8C		3ADA	30	
3AD3	18		3ADB	30	
3AD4	32		3ADC	30	
3AD5	66		3ADD	30	
3AD6	FE		3ADE	3C	
3AD7	00		3ADF	00	
3AE0	C0		3AE8	3C	
3AE1	60		3AE9	0C	
3AE2	30		3AEA	0C	
3AE3	18		3AEB	0C	
3AE4	0C		3AEC	0C	
3AE5	06		3AED	0C	
3AE6	02		3AEE	3C	
3AE7	00		3AEF	00	
3AF0	18		3AF8	00	
3AF1	3C		3AF9	00	
3AF2	7E		3AFA	00	
3AF3	18		3AFB	00	
3AF4	18		3AFC	00	
3AF5	18		3AFD	00	
3AF6	18		3AFE	00	
3AF7	00		3AFF	FF	

CHARACTERS

3B00	30		3B08	00	
3B01	18		3B09	00	
3B02	0C		3B0A	78	
3B03	00		3B0B	0C	
3B04	00		3B0C	7C	
3B05	00		3B0D	CC	
3B06	00		3B0E	76	
3B07	00		3B0F	00	
3B10	E0		3B18	00	
3B11	60		3B19	00	
3B12	7C		3B1A	3C	
3B13	66		3B1B	66	
3B14	66		3B1C	60	
3B15	66		3B1D	66	
3B16	DC		3B1E	3C	
3B17	00		3B1F	00	
3B20	1C		3B28	00	
3B21	0C		3B29	00	
3B22	7C		3B2A	3C	
3B23	CC		3B2B	66	
3B24	CC		3B2C	7E	
3B25	CC		3B2D	60	
3B26	76		3B2E	3C	
3B27	00		3B2F	00	
3B30	1C		3B38	00	
3B31	36		3B39	00	
3B32	30		3B3A	3E	
3B33	78		3B3B	66	
3B34	30		3B3C	66	
3B35	30		3B3D	3E	
3B36	78		3B3E	06	
3B37	00		3B3F	7C	
3B40	E0		3B48	18	
3B41	60		3B49	00	
3B42	6C		3B4A	38	
3B43	76		3B4B	18	
3B44	66		3B4C	18	
3B45	66		3B4D	18	
3B46	E6		3B4E	3C	
3B47	00		3B4F	00	
3B50	06		3B58	E0	
3B51	00		3B59	60	
3B52	0E		3B5A	66	
3B53	06		3B5B	6C	
3B54	06		3B5C	78	
3B55	66		3B5D	6C	
3B56	66		3B5E	E6	
3B57	3C		3B5F	00	

CHARACTERS

3B60	38		3B68	00	
3B61	18		3B69	00	
3B62	18		3B6A	6C	
3B63	18		3B6B	FE	
3B64	18		3B6C	D6	
3B65	18		3B6D	D6	
3B66	3C		3B6E	C6	
3B67	00		3B6F	00	
3B70	00		3B78	00	
3B71	00		3B79	00	
3B72	DC		3B7A	3C	
3B73	66		3B7B	66	
3B74	66		3B7C	66	
3B75	66		3B7D	66	
3B76	66		3B7E	3C	
3B77	00		3B7F	00	
3B80	00		3B88	00	
3B81	00		3B89	00	
3B82	DC		3B8A	76	
3B83	66		3B8B	CC	
3B84	66		3B8C	CC	
3B85	7C		3B8D	7C	
3B86	60		3B8E	0C	
3B87	F0		3B8F	1E	
3B90	00		3B98	00	
3B91	00		3B99	00	
3B92	DC		3B9A	3C	
3B93	76		3B9B	60	
3B94	60		3B9C	3C	
3B95	60		3B9D	06	
3B96	F0		3B9E	7C	
3B97	00		3B9F	00	
3BA0	30		3BA8	00	
3BA1	30		3BA9	00	
3BA2	7C		3BAA	66	
3BA3	30		3BAB	66	
3BA4	30		3BAC	66	
3BA5	36		3BAD	66	
3BA6	1C		3BAE	3E	
3BA7	00		3BAF	00	
3BB0	00		3BB8	00	
3BB1	00		3BB9	00	
3BB2	66		3BBA	C6	
3BB3	66		3BBB	D6	
3BB4	66		3BBC	D6	
3BB5	3C		3BBD	FE	
3BB6	18		3BBE	6C	
3BB7	00		3BBF	00	

CHARACTERS

3BC0	00		3BC8	00	
3BC1	00		3BC9	00	
3BC2	C6		3BCA	66	
3BC3	6C		3BCB	66	
3BC4	38		3BCC	66	
3BC5	6C		3BCD	3E	
3BC6	C6		3BCE	06	
3BC7	00		3BCF	7C	
3BD0	00		3BD8	0E	
3BD1	00		3BD9	18	
3BD2	7E		3BDA	18	
3BD3	4C		3BDB	70	
3BD4	18		3BDC	18	
3BD5	32		3BDD	18	
3BD6	7E		3BDE	0E	
3BD7	00		3BDF	00	
3BE0	18		3BE8	70	
3BE1	18		3BE9	18	
3BE2	18		3BEA	18	
3BE3	18		3BEB	0E	
3BE4	18		3BEC	18	
3BE5	18		3BED	18	
3BE6	18		3BEE	70	
3BE7	00		3BEF	00	
3BF0	76		3BF8	CC	
3BF1	DC		3BF9	33	
3BF2	00		3BFA	CC	
3BF3	00		3BFB	33	
3BF4	00		3BFC	CC	
3BF5	00		3BFD	33	
3BF6	00		3BFE	CC	
3BF7	00		3BFF	33	
3C00	00		3C08	F0	
3C01	00		3C09	F0	
3C02	00		3C0A	F0	
3C03	00		3C0B	F0	
3C04	00		3C0C	00	
3C05	00		3C0D	00	
3C06	00		3C0E	00	
3C07	00		3C0F	00	
3C10	0F		3C18	FF	
3C11	0F		3C19	FF	
3C12	0F		3C1A	FF	
3C13	0F		3C1B	FF	
3C14	00		3C1C	00	
3C15	00		3C1D	00	
3C16	00		3C1E	00	
3C17	00		3C1F	00	

CHARACTERS

3C20	00		3C28	F0	
3C21	00		3C29	F0	
3C22	00		3C2A	F0	
3C23	00		3C2B	F0	
3C24	F0		3C2C	F0	
3C25	F0		3C2D	F0	
3C26	F0		3C2E	F0	
3C27	F0		3C2F	F0	
3C30	0F		3C38	FF	
3C31	0F		3C39	FF	
3C32	0F		3C3A	FF	
3C33	0F		3C3B	FF	
3C34	F0		3C3C	F0	
3C35	F0		3C3D	F0	
3C36	F0		3C3E	F0	
3C37	F0		3C3F	F0	
3C40	00		3C48	F0	
3C41	00		3C49	F0	
3C42	00		3C4A	F0	
3C43	00		3C4B	F0	
3C44	0F		3C4C	0F	
3C45	0F		3C4D	0F	
3C46	0F		3C4E	0F	
3C47	0F		3C4F	0F	
3C50	0F		3C58	FF	
3C51	0F		3C59	FF	
3C52	0F		3C5A	FF	
3C53	0F		3C5B	FF	
3C54	0F		3C5C	0F	
3C55	0F		3C5D	0F	
3C56	0F		3C5E	0F	
3C57	0F		3C5F	0F	
3C60	00		3C68	F0	
3C61	00		3C69	F0	
3C62	00		3C6A	F0	
3C63	00		3C6B	F0	
3C64	FF		3C6C	FF	
3C65	FF		3C6D	FF	
3C66	FF		3C6E	FF	
3C67	FF		3C6F	FF	
3C70	0F		3C78	FF	
3C71	0F		3C79	FF	
3C72	0F		3C7A	FF	
3C73	0F		3C7B	FF	
3C74	FF		3C7C	FF	
3C75	FF		3C7D	FF	
3C76	FF		3C7E	FF	
3C77	FF		3C7F	FF	

CHARACTERS

3C80	00		3C88	18	
3C81	00		3C89	18	
3C82	00		3C8A	18	
3C83	18		3C8B	18	
3C84	18		3C8C	18	
3C85	00		3C8D	00	
3C86	00		3C8E	00	
3C87	00		3C8F	00	
3C90	00		3C98	18	
3C91	00		3C99	18	
3C92	00		3C9A	18	
3C93	1F		3C9B	1F	
3C94	1F		3C9C	0F	
3C95	00		3C9D	00	
3C96	00		3C9E	00	
3C97	00		3C9F	00	
3CA0	00		3CA8	18	
3CA1	00		3CA9	18	
3CA2	00		3CAA	18	
3CA3	18		3CAB	18	
3CA4	18		3CAC	18	
3CA5	18		3CAD	18	
3CA6	18		3CAE	18	
3CA7	18		3CAF	18	
3CB0	00		3CB8	18	
3CB1	00		3CB9	18	
3CB2	00		3CBA	18	
3CB3	0F		3CBB	1F	
3CB4	1F		3CBC	1F	
3CB5	18		3CBD	18	
3CB6	18		3CBE	18	
3CB7	18		3CBF	18	
3CC0	00		3CC8	18	
3CC1	00		3CC9	18	
3CC2	00		3CCA	18	
3CC3	F8		3CCB	F8	
3CC4	F8		3CCC	F0	
3CC5	00		3CCD	00	
3CC6	00		3CCE	00	
3CC7	00		3CCF	00	
3CD0	00		3CD8	18	
3CD1	00		3CD9	18	
3CD2	00		3CDA	18	
3CD3	FF		3CDB	FF	
3CD4	FF		3CDC	FF	
3CD5	00		3CDD	00	
3CD6	00		3CDE	00	
3CD7	00		3CDF	00	

CHARACTERS

3CE0	00		3CE8	18	
3CE1	00		3CE9	18	
3CE2	00		3CEA	18	
3CE3	F0		3CEB	F8	
3CE4	F8		3CEC	F8	
3CE5	18		3CED	18	
3CE6	18		3CEE	18	
3CE7	18		3CEF	18	
3CF0	00		3CF8	18	
3CF1	00		3CF9	18	
3CF2	00		3CFA	18	
3CF3	FF		3CFB	FF	
3CF4	FF		3CFC	FF	
3CF5	18		3CFD	18	
3CF6	18		3CFE	18	
3CF7	18		3CFF	18	
3D00	10		3D08	0C	
3D01	38		3D09	18	
3D02	6C		3D0A	30	
3D03	C6		3D0B	00	
3D04	00		3D0C	00	
3D05	00		3D0D	00	
3D06	00		3D0E	00	
3D07	00		3D0F	00	
3D10	66		3D18	3C	
3D11	66		3D19	66	
3D12	00		3D1A	60	
3D13	00		3D1B	F8	
3D14	00		3D1C	60	
3D15	00		3D1D	66	
3D16	00		3D1E	FE	
3D17	00		3D1F	00	
3D20	38		3D28	7E	
3D21	44		3D29	F4	
3D22	BA		3D2A	F4	
3D23	A2		3D2B	74	
3D24	BA		3D2C	34	
3D25	44		3D2D	34	
3D26	38		3D2E	34	
3D27	00		3D2F	00	
3D30	1E		3D38	18	
3D31	30		3D39	18	
3D32	38		3D3A	0C	
3D33	6C		3D3B	00	
3D34	38		3D3C	00	
3D35	18		3D3D	00	
3D36	F0		3D3E	00	
3D37	00		3D3F	00	

CHARACTERS

3D40	40		3D48	40	
3D41	C0		3D49	C0	
3D42	44		3D4A	4C	
3D43	4C		3D4B	52	
3D44	54		3D4C	44	
3D45	1E		3D4D	08	
3D46	04		3D4E	1E	
3D47	00		3D4F	00	
3D50	E0		3D58	00	
3D51	10		3D59	18	
3D52	62		3D5A	18	
3D53	16		3D5B	7E	
3D54	EA		3D5C	18	
3D55	0F		3D5D	18	
3D56	02		3D5E	7E	
3D57	00		3D5F	00	
3D60	18		3D68	00	
3D61	18		3D69	00	
3D62	00		3D6A	00	
3D63	7E		3D6B	7E	
3D64	00		3D6C	06	
3D65	18		3D6D	06	
3D66	18		3D6E	00	
3D67	00		3D6F	00	
3D70	18		3D78	18	
3D71	00		3D79	00	
3D72	18		3D7A	18	
3D73	30		3D7B	18	
3D74	66		3D7C	18	
3D75	66		3D7D	18	
3D76	3C		3D7E	18	
3D77	00		3D7F	00	
3D80	00		3D88	7C	
3D81	00		3D89	C6	
3D82	73		3D8A	C6	
3D83	DE		3D8B	FC	
3D84	CC		3D8C	C6	
3D85	DE		3D8D	C6	
3D86	73		3D8E	F8	
3D87	00		3D8F	C0	
3D90	00		3D98	3C	
3D91	66		3D99	60	
3D92	66		3D9A	60	
3D93	3C		3D9B	3C	
3D94	66		3D9C	66	
3D95	66		3D9D	66	
3D96	3C		3D9E	3C	
3D97	00		3D9F	00	

CHARACTERS

3DA0	00		3DA8	38	
3DA1	00		3DA9	6C	
3DA2	1E		3DAA	C6	
3DA3	30		3DAB	FE	
3DA4	7C		3DAC	C6	
3DA5	30		3DAD	6C	
3DA6	1E		3DAE	38	
3DA7	00		3DAF	00	
3DB0	00		3DB8	00	
3DB1	C0		3DB9	00	
3DB2	60		3DBA	66	
3DB3	30		3DBB	66	
3DB4	38		3DBC	66	
3DB5	6C		3DBD	7C	
3DB6	C6		3DBE	60	
3DB7	00		3DBF	60	
3DC0	00		3DC8	00	
3DC1	00		3DC9	00	
3DC2	00		3DCA	00	
3DC3	FE		3DCB	7E	
3DC4	6C		3DCC	D8	
3DC5	6C		3DCD	D8	
3DC6	6C		3DCE	70	
3DC7	00		3DCF	00	
3DD0	03		3DD8	03	
3DD1	06		3DD9	06	
3DD2	0C		3DDA	0C	
3DD3	3C		3ddb	66	
3DD4	66		3DDC	66	
3DD5	3C		3DDD	3C	
3DD6	60		3DDE	60	
3DD7	C0		3DDF	C0	
3DE0	00		3DE8	00	
3DE1	E6		3DE9	00	
3DE2	3C		3DEA	66	
3DE3	18		3DEB	C3	
3DE4	38		3DEC	DB	
3DE5	6C		3DED	DB	
3DE6	C7		3DEE	7E	
3DE7	00		3DEF	00	
3DF0	FE		3DF8	00	
3DF1	C6		3DF9	7C	
3DF2	60		3DFA	C6	
3DF3	30		3DFB	C6	
3DF4	60		3DFC	C6	
3DF5	C6		3DFD	6C	
3DF6	FE		3DFE	EE	
3DF7	00		3DFE	00	

CHARACTERS

3E00	18		3E08	18	
3E01	30		3E09	0C	
3E02	60		3E0A	06	
3E03	C0		3E0B	03	
3E04	80		3E0C	01	
3E05	00		3E0D	00	
3E06	00		3E0E	00	
3E07	00		3E0F	00	
3E10	00		3E18	00	
3E11	00		3E19	00	
3E12	00		3E1A	00	
3E13	01		3E1B	80	
3E14	03		3E1C	C0	
3E15	06		3E1D	60	
3E16	0C		3E1E	30	
3E17	18		3E1F	18	
3E20	18		3E28	18	
3E21	3C		3E29	0C	
3E22	66		3E2A	06	
3E23	C3		3E2B	03	
3E24	81		3E2C	03	
3E25	00		3E2D	06	
3E26	00		3E2E	0C	
3E27	00		3E2F	18	
3E30	00		3E38	18	
3E31	00		3E39	30	
3E32	00		3E3A	60	
3E33	81		3E3B	C0	
3E34	C3		3E3C	C0	
3E35	66		3E3D	60	
3E36	3C		3E3E	30	
3E37	18		3E3F	18	
3E40	18		3E48	18	
3E41	30		3E49	0C	
3E42	60		3E4A	06	
3E43	C1		3E4B	83	
3E44	83		3E4C	C1	
3E45	06		3E4D	60	
3E46	0C		3E4E	30	
3E47	18		3E4F	18	
3E50	18		3E58	C3	
3E51	3C		3E59	E7	
3E52	66		3E5A	7E	
3E53	C3		3E5B	3C	
3E54	C3		3E5C	3C	
3E55	66		3E5D	7E	
3E56	3C		3E5E	E7	
3E57	18		3E5F	C3	

CHARACTERS

3E60	03		3E68	C0	
3E61	07		3E69	E0	
3E62	0E		3E6A	70	
3E63	1C		3E6B	38	
3E64	38		3E6C	1C	
3E65	70		3E6D	0E	
3E66	E0		3E6E	07	
3E67	C0		3E6F	03	
3E70	CC		3E78	AA	
3E71	CC		3E79	55	
3E72	33		3E7A	AA	
3E73	33		3E7B	55	
3E74	CC		3E7C	AA	
3E75	CC		3E7D	55	
3E76	33		3E7E	AA	
3E77	33		3E7F	55	
3E80	FF		3E88	03	
3E81	FF		3E89	03	
3E82	00		3E8A	03	
3E83	00		3E8B	03	
3E84	00		3E8C	03	
3E85	00		3E8D	03	
3E86	00		3E8E	03	
3E87	00		3E8F	03	
3E90	00		3E98	C0	
3E91	00		3E99	C0	
3E92	00		3E9A	C0	
3E93	00		3E9B	C0	
3E94	00		3E9C	C0	
3E95	00		3E9D	C0	
3E96	FF		3E9E	C0	
3E97	FF		3E9F	C0	
3EA0	FF		3EA8	FF	
3EA1	FE		3EA9	7F	
3EA2	FC		3EAA	3F	
3EA3	F8		3EAB	1F	
3EA4	F0		3EAC	0F	
3EA5	E0		3EAD	07	
3EA6	C0		3EAE	03	
3EA7	80		3EAF	01	
3EB0	01		3EB8	80	
3EB1	03		3EB9	C0	
3EB2	07		3EBA	E0	
3EB3	0F		3EBB	F0	
3EB4	1F		3EBC	F8	
3EB5	3F		3EBD	FC	
3EB6	7F		3EBE	FE	
3EB7	FF		3EBF	FF	

CHARACTERS

3EC0	AA		3EC8	0A	
3EC1	55		3EC9	05	
3EC2	AA		3ECA	0A	
3EC3	55		3ECB	05	
3EC4	00		3ECC	0A	
3EC5	00		3ECD	05	
3EC6	00		3ECE	0A	
3EC7	00		3ECF	05	
3ED0	00		3ED8	A0	
3ED1	00		3ED9	50	
3ED2	00		3EDA	A0	
3ED3	00		3EDB	50	
3ED4	AA		3EDC	A0	
3ED5	55		3EDD	50	
3ED6	AA		3EDE	A0	
3ED7	55		3EDF	50	
3EE0	AA		3EE8	AA	
3EE1	54		3EE9	55	
3EE2	A8		3EEA	2A	
3EE3	50		3EEB	15	
3EE4	A0		3EEC	0A	
3EE5	40		3EED	05	
3EE6	80		3EEE	02	
3EE7	00		3EEF	01	
3EF0	01		3EF8	00	
3EF1	02		3EF9	80	
3EF2	05		3EFA	40	
3EF3	0A		3EFB	A0	
3EF4	15		3EFC	50	
3EF5	2A		3EFD	A8	
3EF6	55		3EFE	54	
3EF7	AA		3EFF	AA	
3F00	7E		3F08	7E	
3F01	FF		3F09	FF	
3F02	99		3F0A	99	
3F03	FF		3F0B	FF	
3F04	BD		3F0C	C3	
3F05	C3		3F0D	BD	
3F06	FF		3F0E	FF	
3F07	7E		3F0F	7E	
3F10	38		3F18	10	
3F11	38		3F19	38	
3F12	FE		3F1A	7C	
3F13	FE		3F1B	FE	
3F14	FE		3F1C	7C	
3F15	10		3F1D	38	
3F16	38		3F1E	10	
3F17	00		3F1F	00	

































CHARACTERS

3F20	6C		3F28	10	
3F21	FE		3F29	38	
3F22	FE		3F2A	7C	
3F23	FE		3F2B	FE	
3F24	7C		3F2C	FE	
3F25	38		3F2D	10	
3F26	10		3F2E	38	
3F27	00		3F2F	00	
3F30	00		3F38	00	
3F31	3C		3F39	3C	
3F32	66		3F3A	7E	
3F33	C3		3F3B	FF	
3F34	C3		3F3C	FF	
3F35	66		3F3D	7E	
3F36	3C		3F3E	3C	
3F37	00		3F3F	00	
3F40	00		3F48	00	
3F41	7E		3F49	7E	
3F42	66		3F4A	7E	
3F43	66		3F4B	7E	
3F44	66		3F4C	7E	
3F45	66		3F4D	7E	
3F46	7E		3F4E	7E	
3F47	00		3F4F	00	
3F50	0F		3F58	3C	
3F51	07		3F59	66	
3F52	0D		3F5A	66	
3F53	78		3F5B	66	
3F54	CC		3F5C	3C	
3F55	CC		3F5D	18	
3F56	CC		3F5E	7E	
3F57	78		3F5F	18	
3F60	0C		3F68	18	
3F61	0C		3F69	1C	
3F62	0C		3F6A	1E	
3F63	0C		3F6B	1B	
3F64	0C		3F6C	18	
3F65	3C		3F6D	78	
3F66	7C		3F6E	F8	
3F67	38		3F6F	70	
3F70	99		3F78	10	
3F71	5A		3F79	38	
3F72	24		3F7A	38	
3F73	C3		3F7B	38	
3F74	C3		3F7C	38	
3F75	24		3F7D	38	
3F76	5A		3F7E	7C	
3F77	99		3F7F	D6	

CHARACTERS

3F80	18		3F88	18	
3F81	3C		3F89	18	
3F82	7E		3F8A	18	
3F83	FF		3F8B	18	
3F84	18		3F8C	FF	
3F85	18		3F8D	7E	
3F86	18		3F8E	3C	
3F87	18		3F8F	18	
3F90	10		3F98	08	
3F91	30		3F99	0C	
3F92	70		3F9A	0E	
3F93	FF		3F9B	FF	
3F94	FF		3F9C	FF	
3F95	70		3F9D	0E	
3F96	30		3F9E	0C	
3F97	10		3F9F	08	
3FA0	00		3FA8	00	
3FA1	00		3FA9	00	
3FA2	18		3FAA	FF	
3FA3	3C		3FAB	FF	
3FA4	7E		3FAC	7E	
3FA5	FF		3FAD	3C	
3FA6	FF		3FAE	18	
3FA7	00		3FAF	00	
3FB0	80		3FB8	02	
3FB1	E0		3FB9	0E	
3FB2	F8		3FBA	3E	
3FB3	FE		3FBB	FE	
3FB4	F8		3FBC	3E	
3FB5	E0		3FBD	0E	
3FB6	80		3FBE	02	
3FB7	00		3FBF	00	
3FC0	38		3FC8	38	
3FC1	38		3FC9	38	
3FC2	92		3FCA	10	
3FC3	7C		3FCB	FE	
3FC4	10		3FCC	10	
3FC5	28		3FCD	28	
3FC6	28		3FCE	44	
3FC7	28		3FCF	82	
3FD0	38		3FD8	38	
3FD1	38		3FD9	38	
3FD2	12		3FDA	90	
3FD3	7C		3FDB	7C	
3FD4	90		3FDC	12	
3FD5	28		3FDD	28	
3FD6	24		3FDE	48	
3FD7	22		3FDF	88	

CHARACTERS

3FE0	00		3FE8	3C	
3FE1	3C		3FE9	FF	
3FE2	18		3FEA	FF	
3FE3	3C		3FEB	18	
3FE4	3C		3FEC	0C	
3FE5	3C		3FED	18	
3FE6	18		3FEE	30	
3FE7	00		3FEF	18	
3FF0	18		3FF8	00	
3FF1	3C		3FF9	24	
3FF2	7E		3FFA	66	
3FF3	18		3FFB	FF	
3FF4	18		3FFC	66	
3FF5	7E		3FFD	24	
3FF6	3C		3FFE	00	
3FF7	18		3FFF	00	

LE BASIC

3.1 L'interpréteur Basic du CPC

Le CPC dispose d'un interpréteur Basic rapide et puissant qui est logé dans une Rom de 16 K. Il occupe la zone d'adresses &C000 à &FFFF, parallèlement à la Ram écran. Pour le programme Basic et pour les variables Basic, la zone de &0170 à &AB80 est disponible, ce qui représente 43534 octets.

L'interpréteur soutient presque toutes les possibilités offertes par l'électronique et le système d'exploitation de l'ordinateur. Cela comprend notamment la sortie sur écran avec jusqu'à 8 fenêtres, le graphisme haute résolution, le son ainsi que le traitement d'événement. Il est ainsi pour la première fois possible de faire exécuter en Basic plusieurs tâches parallèlement. L'interpréteur Basic offre en outre une arithmétique avec des nombres entiers de 16 bits (zone de valeurs de -32768 à 32767) et une arithmétique avec virgule flottante avec puissance de deux sur 8 bits et une mantisse de 32 bits qui garantit une précision de 9 décimales pour une zone de valeurs de +- 1E-39 à +- 1E+38.

L'arithmétique entière ou l'arithmétique à virgule flottante ne font cependant pas partie de l'interpréteur Basic mais de la Rom du système d'exploitation (adresses &2E18 à &37FF). Elles sont appelées comme les autres fonctions du système d'exploitation à travers la table de saut qui se trouve dans le haut de la Ram (&BB00 à &BDF1) et qui peut être modifiée en cas de besoin.

L'interpréteur Basic permet également la création, l'édition (examen/modification) et l'exécution aisées de programmes. La création de programmes est en effet facilitée par l'instruction AUTO, l'édition par l'instruction EDIT qui, grâce à la puissance du système d'exploitation, est à peine moins maniable que l'éditeur plein écran ainsi que par les instructions RENUM, MERGE et DELETE. L'exécution des programmes est également facilitée par des instructions puissantes. Par exemple l'instruction ON ERROR GOTO permet le traitement des erreurs. L'instruction DEFtype permet de définir le type d'une variable, l'instruction ERASE permet une suppression sélective de tableaux. Il est encore possible d'entrer et de faire sortir les nombres comme des nombres décimaux, binaires ou hexadécimaux ainsi que d'utiliser des fonctions

qu'on a soi-même définies, fonctions qui peuvent comporter plusieurs arguments. Enfin les structures de programme telles que IF ... THEN ... ELSE, FOR ... NEXT et WHILE ... WEND sont un autre aspect très important de la puissance du Basic du CPC. Il est également possible en Basic de réaffecter es touches du clavier, de définir les fonctions des touches de fonction ou de définir des caractères qui apparaîtront à l'écran. Il ne manque ni l'instruction TRACE ni une très complète instruction PRINT USING.

Après ce bref aperçu, nous allons nous pencher de plus près sur l'entrée et le stockage des lignes de Basic, ainsi que sur l'exécution des programmes par l'interpréteur Basic. Ces informations vous permettront non seulement de pouvoir tirer le maximum de votre interpréteur Basic mais également d'écrire vos propres extensions du Basic. Nous vous donnerons plus loin quelques exemples d'extensions du Basic.

L'entrée de lignes Basic

Lorsque vous entrez une ligne Basic, elle est d'abord placée dans un buffer de 256 octets qui se trouve aux adresses &ACA4 à &ADA3. L'entrée y figure en clair, non codée. Si la ligne commence par un numéro, celui-ci est converti en un nombre binaire de 16 bits et placé dans un second buffer destiné à recevoir la ligne traitée. Ce buffer comprend 300 caractères et il se trouve avant le programme Basic, aux adresses &40 à &16F. La ligne entrée est alors examinée pour voir si elle comporte des mots-clé Basic. Ces mots-clés sont remplacés par un octet appelé token. Par exemple 'AFTER' devient le token &80. Les tokens de tous les mots d'instruction et des opérateurs Basic tels que '=' ou 'AND' ont des valeurs supérieures à 127, c'est-à-dire que leur bit 7 est mis. Les fonctions Basic comme EXP ou ROUND ont des tokens compris entre 0 et &7F. Pour les distinguer des caractères ASCII normaux, ils sont marqués par un &FF les précédant. Le double-point servant à séparer entre elles deux instructions est représenté par le code &01, la fin d'une ligne est marquée par un &00. Si une suite de lettres n'a pu être identifiée comme étant une instruction ou une fonction, elle est traitée comme étant le nom d'une variable. Un nom de variable peut comprendre jusqu'à 40 caractères qui sont tous significatifs. Aucune différence n'est faite entre les majuscules et les minuscules. Supposons que nous ayons entré la ligne suivante:

10 start=77

Après le numéro de ligne seront placées les valeurs:

&0D &00 &00 &73 &74 &61 &72 &F4 &EF &19 &4D &00

Le &0D indique qu'il s'agit d'une variable sans marque de type. Ensuite viennent deux 0 sur lesquels nous reviendrons plus tard. Puis vient le nom de la variable, les codes ASCII pour s, t, a et r. Pour la dernière lettre, 't', &80 est ajouté au code ASCII &74 (le bit supérieur est mis) et nous obtenons &F4. Le code &EF est le token pour '='. Le code &19 qui suit indique une constante à un octet: &4D est la valeur de cette constante (=77 en décimal). Le zéro qui termine marque la fin de la ligne.

Avant le numéro de ligne, il y a encore deux octets qui indiquent la longueur de la ligne:

&12 &00 &0A &00

La ligne comporte donc &12+256*&00 soient 18 octets et elle porte le numéro de ligne &0A+256*&00, soit 10.

Vous voyez donc qu'au contraire de ce qui est le cas avec d'autres interpréteurs Basic, les constantes ne sont pas placées dans le texte du programme sous forme de textes ASCII, mais sous la forme de leur traduction binaire. Ceci présente un avantage décisif. La conversion du format ASCII au format binaire prend en effet du temps. Avec la technique utilisée sur le CPC, cette conversion ne s'effectue qu'une seule fois, lors de l'entrée de la ligne et elle n'a donc pas à être effectuée chaque fois que la ligne est exécutée. Il en découle un gain de vitesse dans l'exécution des programmes qui n'est pas négligeable.

Le CPC connaît d'autre part toute une série de constantes numériques qui sont désignées par un token particulier. Les constantes qui ne comprennent par exemple qu'un seul chiffre, soient les nombres de 0 à 9 sont ainsi codées avec les tokens &0E à &17. Elles n'occupent ainsi qu'un octet dans le texte du programme. Nous avons déjà rencontré le token &19 qui marque les valeurs numériques d'un octet. Pour les valeurs entières sur deux octets, il y a trois tokens différents, suivant que la constante

a été entrée sous la forme décimale, binaire ou hexadécimale. La valeur de la constante est toujours stockée de la même façon avec un octet faible et un octet fort.

- &1A valeur sur deux octets, décimal
- &1B valeur sur deux octets, binaire
- &1C valeur sur deux octets, hexadécimal

S'il ne s'agit pas d'un nombre entier ou si sa valeur est supérieure à 32767, le nombre est stocké sous la forme d'une valeur à virgule flottante qui est désignée par le token &1F. Le token est suivi de la valeur à virgule flottante sur 5 octets. Nous reviendrons plus tard sur les valeurs à virgule flottante.

Dans ce contexte, les numéros de ligne ont une situation particulière lorsqu'ils suivent par exemple des instructions telles que GOTO, GOSUB ou RUN. Ils sont également stockés sous la forme binaire, mais ils sont désignés par le token &1E.

Lorsqu'un programme est exécuté et qu'il rencontre par exemple une instruction GOTO, il lit alors le numéro de ligne et il doit rechercher cette ligne dans tout le programme. Sur des programmes de taille importante, cela peut durer assez longtemps. Les instructions GOTO et GOSUB sont souvent utilisées dans des boucles qui sont parcourues des centaines ou des milliers de fois. Dans ce cas, le temps de recherche des numéros de ligne peut représenter une part importante du temps d'exécution du programme. L'interpréteur Basic du CPC n'effectue cette recherche de ligne qu'une seule fois. En effet, une fois qu'il a trouvé la ligne recherchée, il remplace le numéro de ligne figurant à la suite de l'instruction GOTO par l'adresse de cette ligne qu'il vient de trouver. Pour qu'il puisse faire la différence entre un numéro de ligne et une adresse de ligne, il remplace le token &1E par le token &1D, qui est le token pour les adresses de ligne. Si la même instruction GOTO est exécutée une seconde fois, l'interpréteur trouve directement l'adresse à laquelle le programme doit sauter, ce qui permet bien sûr de gagner beaucoup de temps.

Cette technique crée cependant quelques difficultés pour les instructions qui utilisent le numéro de ligne en tant que tel. Lorsque l'instruction LIST doit par exemple sortir le numéro de ligne, c'est le numéro de ligne

qu'elle doit indiquer et non l'adresse de la ligne. Ce problème est cependant très facilement résolu. En effet lorsque l'adresse de la ligne est connue, il est facile d'aller y rechercher le numéro de ligne puisque, comme nous l'avons vu, le numéro de ligne est stocké dans la ligne. Lorsque des lignes sont supprimées ou que d'autres lignes sont ajoutées, les adresses de ligne doivent être remplacées par les numéros de ligne car de telles opérations entraînent bien sûr une modification des adresses de ligne. Cela ne présente cependant d'inconvénient que pour l'entrée et la sortie de lignes de programmes. Ce petit inconvénient est cependant largement compensé par la vitesse nettement plus grande d'exécution des programmes.

L'exécution des programmes par l'interpréteur Basic

L'exécution d'une instruction par l'interpréteur Basic se présente, en simplifiant un peu, de la façon suivante. Chaque ligne de programme commence, comme nous l'avons dit, par la longueur de la ligne et le numéro de ligne. Ensuite vient l'instruction Basic proprement dite. L'interpréteur examine maintenant s'il s'agit d'un token d'instruction, dont la valeur est toujours comprise entre &80 et &DC. Si c'est le cas, il utilise ce token comme pointeur d'une table qui contient les adresses de toutes les instructions Basic. L'instruction Basic est alors exécutée comme un sous-programme. On revient ensuite à ce qu'on appelle la boucle de l'interpréteur. Si l'instruction ne commençait cependant pas par un token d'instruction, on saute à l'instruction LET.

La partie la plus importante de l'interpréteur Basic est certainement le calcul des expressions. Le CPC distingue à cet égard trois types d'expressions: entières, à virgule flottante et chaînes de caractères. Lorsque par exemple une affectation de valeur à une variable est exécutée ou lorsque le paramètre d'une instruction doit être calculé, une routine est appelée qui calcule l'expression et qui fournit la valeur ainsi que le type de l'expression. Le type de variable peut avoir trois valeurs différentes:

- 2 entier
- 3 chaîne
- 5 virgule flottante

Ce numéro de type donne en même temps la longueur de la variable. Pour une chaîne, c'est ce qu'on appelle le Descriptor qui contient la longueur et l'adresse de la chaîne (voyez également le chapitre sur le pointeur de variable): Si cependant le type d'une expression est différent du type d'une variable à laquelle cette expression doit être affectée, une conversion de type est tentée, mais seulement entre les deux types numériques entier et à virgule flottante. Cette conversion prend bien sûr un certain temps et il est donc préférable d'employer des variables entières lorsque c'est possible. L'expérience révèle en effet que le type entier convient dans 90 % des cas. Non seulement le type entier évite les conversions de types, mais l'arithmétique entière est en outre nettement plus rapide que l'arithmétique à virgule flottante. Cette remarque vaut particulièrement pour les variables de comptage utilisées par exemple dans les boucles FOR...NEXT.

Par contre, si vous tentez d'affecter une expression du type chaîne de caractères à une variable numérique ou vice versa, le message d'erreur 'Type mismatch' sera sorti. La conversion de chaîne de caractères à numérique et vice versa n'est possible qu'avec les fonctions VAL et STR\$.

3.2 La pile Basic

Une pile ou mémoire de pile (stack) permet de stocker des données suivant le principe 'Last in - First out' (dernier entré - premier sorti). Le processeur utilise à cet effet la zone de mémoire commençant en &C000. Avant chaque entrée, le pointeur de pile (stack pointer) est décrémenté. Lorsqu'on retire des données de la pile, le pointeur de pile est incrémenté immédiatement après. La pile du processeur sert par exemple à placer les adresses de retour lors de l'appel de sous-programmes et elle permet, grâce au principe d'accès utilisé, de réaliser une imbrication des sous-programmes.

L'interpréteur Basic a également besoin d'une pile pour stocker les paramètres des appels par GOSUB ou des boucles FOR-NEXT et WHILE-WEND. Seule une pile permet en effet de réaliser une imbrication de ces différentes structures de programme. On utilise pas à cet effet la pile du processeur car il existe une pile Basic de 512 octets qui commence à l'adresse &AE8B. Au contraire de la pile du processeur, cette pile croît vers les adresses plus élevées, au fur et à mesure que le nombre d'entrées augmente, jusqu'à l'adresse limite &B08A. Les cases mémoire &B08B et &B08C font office de pointeur de pile.

Voyons d'abord quels paramètres sont placés sur la pile pour une instruction GOSUB:

&00/&01	marque du type de GOSUB
Lo	Adresse de l'instruction suivant
Hi	l'instruction GOSUB
Lo	Adresse de la ligne de
Hi	l'instruction GOSUB
&06	Taille de l'entrée sur la pile

Un octet est donc tout d'abord placé sur la pile qui détermine le type de l'instruction GOSUB. Pour un GOSUB normal, il s'agit d'un octet nul. S'il s'agit cependant de l'appel d'un sous-programme par une instruction AFTER ou EVERY, c'est un 1 qui sera placé sur la pile. Viennent ensuite l'adresse de la prochaine instruction après l'instruction GOSUB ainsi que

l'adresse de la ligne dans laquelle figure l'instruction GOSUB. Pour que l'entrée sur la pile puisse être identifiée à nouveau lorsque l'instruction RETURN sera exécutée, un octet est encore placé sur la pile qui indique la longueur de l'entrée sur la pile et indique ainsi implicitement qu'il s'agit d'un enregistrement concernant une instruction GOSUB.

Les données pour une boucle WHILE-WEND sont placées de façon similaires:

Lo Adresse de la ligne de
Hi l'instruction WHILE

Lo Adresse de
Hi l'instruction WEND

Lo Adresse de
Hi la condition WHILE

&07 Taille de l'entrée sur la pile

L'entrée comporte donc trois adresses et un octet d'identification qui vaut 7 et qui indique également le nombre d'octets de données entrés sur la pile.

Les choses se compliquent un peu avec la boucle FOR-NEXT. On fait ici une distinction selon que la variable de comptage est du type entier ou du type réel. Dans le premier cas, non seulement le temps d'exécution est plus court, mais la place occupée sur la pile est en outre moindre. Considérons tout d'abord la structure d'une boucle de type entier.

Lo Adresse de la
Hi variable de comptage

Lo Valeur finale de la
Hi variable de comptage

Lo Valeur STEP
Hi

Sgn Signe de la valeur STEP

Lo	Adresse de
Hi	l'instruction FOR
Lo	Adresse de la ligne de
Hi	l'instruction FOR
Lo	Adresse de
Hi	l'instruction NEXT
Lo	Adresse de la ligne
Hi	de l'instruction NEXT
&10	Taille de l'entrée sur la pile

L'entrée sur la pile pour une boucle FOR-NEXT avec variable entière est donc longue de 16 octets. Si une boucle utilise une variable de comptage de type réel, ce sont 22 octets qui seront placés sur la pile.

Lo	Adresse de la
Hi	variable de comptage
Valeur à virgule flottante sur 5 octets	Valeur finale de la variable de comptage
Valeur à virgule flottante sur 5 octets	Valeur STEP
Sgn	Signe de la valeur STEP
Lo	Adresse de
Hi	l'instruction FOR
Lo	Adresse de la ligne de
Hi	l'instruction FOR
Lo	Adresse de
Hi	l'instruction NEXT

Lo	Adresse de la ligne
Hi	de l'instruction NEXT
&16	Taille de l'entrée sur la pile

Outre le stockage des structures de programme, la pile Basic sert également au stockage d'expressions provisoires pour les calculs numériques, par exemple pour le calcul d'expressions imbriquées entre parenthèses et pour réaliser une hiérarchie pour les opérateurs arithmétiques et logiques.

3.3 Les vecteurs Basic

Si vous voulez réaliser vos propres instructions ou fonctions Basic, vous pouvez utiliser des Roms d'extension ou RSX mais vous pouvez également utiliser ce qu'on appelle les vecteurs Basic.

L'interpréteur Basic utilise dans tous les endroits stratégiques un sous-programme de la Ram qui ne se compose normalement que d'une instruction RET et qui n'influence donc pas le cours des opérations. Par exemple, si une instruction doit être exécutée, on teste d'abord si l'instruction commence par un token d'instruction valable. Si c'est bien le cas, l'adresse de l'instruction correspondante est calculée grâce au token et on saute à cette adresse. Si par contre aucune instruction valable n'a été identifiée, un SYNTAX ERROR est annoncé. Avant que ne soit cependant sorti le message d'erreur -et c'est là qu'est l'astuce- la routine dont nous parlons plus haut est appelée. Celle-ci ne se compose normalement que d'une instruction RET et elle est donc normalement immédiatement abandonnée. Si vous voulez donc intégrer votre propre instruction, il vous suffit de remplacer cette instruction RET qui figure en Ram par un saut à votre propre routine qui contrôlera la validité de la nouvelle instruction et l'exécutera.

Cette méthode est plus souple que la méthode RSX -on n'est pas limité aux arguments entiers (voir chapitre 3.5.2)- et son exécution est plus rapide puisqu'il n'est pas nécessaire de rechercher le mot d'instruction correcte dans la table de toutes les extensions.

Le tableau suivant contient les adresses des 9 routines qui peuvent être manipulées par l'utilisateur. L'interpréteur Basic n'appelle pas ici une routine, mais saute directement à la sortie des erreurs (voir listing du Basic, adresse &D078).

AC01-AC03	Branchement pour mode READY
AC04-AC06	Branchement pour entrée ERROR
AC07-AC09	Branchement pour exécuter instruction
AC0A-AC0C	Branchement pour calcul de fonction
AC0D-AC0F	Branchement pour aller chercher constante (inutilisé)
AC10-AC12	Branchement pour entrée, convertir ligne en token
AC13-AC15	Branchement pour sortie, lister token
AC16-AC18	Branchement pour entrée, conversion de chiffres

AC19-AC1B Branchement pour opérateurs

L'exemple suivant montre un exemple d'application de ces vecteurs. On utilise pour cela l'instruction SWAP dont le token existe déjà pour échanger deux variables entre elles.

```

;Swap de variables
;L.E. 15/12/84

00E7      SWAP   EQU   &E7;token SWAP
CA94      ERROR  EQU   &CA94;sortie erreurs
000D      MISM   EQU   13;'TYPE MISMATCH'
D686      GETVAR EQU   &D686;chercher variable
DD37      CHECK  EQU   &DD37;examiner caractère
DD3F      BLANK  EQU   &DD3F;ignorer les espaces

AC07      ORG    &AC07

AC07 C30080      JP    SWPNEW;détourner le vecteur

8000      ORG    &8000

8000 FECE      SWPNEW CP    SWAP*2 & &FF;token SWAP?
8002 C0        RET    NZ;'SYNTAX ERROR'
8003 D1        POP    DE;retirer adresse de retour de pile
8004 CD3FDD    CALL   BLANK
8007 CD86D6    CALL   GETVAR;chercher variable
800A CD37DD    CALL   CHECK
800D 2C        DEFB   ",,";tester si virgule
800E ED534C80  LD     (VAR1),DE
8012 C5        PUSH   BC;et ranger le type
8013 CD86D6    CALL   GETVAR
8016 ED534E80  LD     (VAR2),DE
801A 79        LD     A,C
801B C1        POP    BC
801C B9        CP     C;comparer les types
801D 2805      JR     Z,TYPOK

801F 1E0D      LD     E,MISM
8021 C394CA    JP     ERROR

```

8024	E5	TYPOK	PUSH	HL;ranger le pointeur de programme
8025	0600		LD	B,0
8027	2A4E80		LD	HL,(VAR2)
802A	114780		LD	DE,TEMP
802D	C5		PUSH	BC
802E	EDB0		LDIR	,variable 2 => TEMP
8030	C1		POP	BC
8031	2A4C80		LD	HL,(VAR1)
8034	ED5B4E80		LD	DE,(VAR2)
8038	C5		PUSH	BC
8039	ED80		LDIR	,variable 1 => variable 2
803B	C1		POP	BC
803C	214780		LD	HL,TEMP
803F	ED5B4C80		LD	DE,(VAR1)
8043	EDB0		LDIR	,TEMP => variable 1
8045	E1		POP	HL;retirer le pointeur de programme
8046	C9		RET	
8047		TEMP	DEFS	5;mémoire provisoire
804C		VAR1	DEFS	2;adresse variable 1
804E		VAR2	DEFS	2;adresse variable 2

La case mémoire &AC00 joue également un rôle. Normalement c'est un 0 qui y figure. La conséquence en est qu'une ligne Basic est reçue comme elle a été entrée. Si toutefois on charge dans cette case mémoire une valeur non nulle, les espaces superflus de la ligne entrée seront ignorés et ne seront pas stockés avec la ligne.

Si vous entrez par exemple la ligne suivante:

```
10 FOR i=1 TO 100: PRINT "Bonjour" : NEXT
```

vous obtiendrez:

```
10 FOR i=1 TO 100:PRINT"Bonjour":NEXT
```

Cette fonction peut être utile lorsque la place en mémoire encore disponible se fait rare. Le programme est ainsi comprimé autant qu'il est possible. L'inconvénient de cette méthode est que le programme risque de perdre en clarté et lisibilité. Les structures de programme dont voici un

exemple peuvent en effet être repérées plus difficilement.

```
10FORI=1 TO 100
20 PRINT"Bonjour"
30NEXT
```

3.4 La Ram Basic

Voici une liste vous présentant la signification des adresses de la Ram utilisées par l'interpréteur Basic.

AB80-ABFF Matrice de caractère pour CHR\$(240) à CHR\$(255)
AC00 Ignorer le flag pour espace supplémentaires
AC01-AC03 Branchement pour mode READY
AC04-AC06 Branchement pour entrée ERROR
AC07-AC09 Branchement pour exécuter instruction
AC0A-AC0C Branchement pour calcul de fonction
AC0D-AC0F Branchement pour aller chercher constante (inutilisé)
AC10-AC12 Branchement pour entrée, convertir ligne en token
AC13-AC15 Branchement pour sortie, lister token
AC16-AC18 Branchement pour entrée, conversion de chiffres
AC19-AC1B Branchement pour opérateurs
AC1C Flag pour mode AUTO
AC1D-AC1E Numéro de ligne pour AUTO
AC1F-AC20 Incrément pour AUTO
AC21 Numéro stream actuel
AC22 Canal d'entrée
AC23 Position actuelle imprimante
AC24 WIDTH
AC25 actuelle position sur cassette
AC26 Flag pour premier parcours FOR-NEXT
AC27-AC2B Mémoire provisoire pour variable FOR
AC2C-AC2D Adresse de l'instruction NEXT correspondante
AC2E-AC2F Adresse de l'instruction WEND correspondante
AC34-AC35 Adresse ON BREAK
AC36-AC37
AC38-AC43 Sound-Queue 0
AC44-AC4F Sound-Queue 1
AC50-AC5B Sound-Queue 2
AC5C-AC6D Event-block 0
AC6E-AC7F Event-block 1
AC80-AC91 Event-block 2
AC92-ACA3 Event-block 3
ACA4-ADA3 Buffer d'entrée
ADA6-ADA7 Adresse de la ligne ERROR
ADA8-ADA9 Pointeur de programme après ERROR

ADAA Numéro ERROR
 ADAB-ADAC Pointeur de programme après interruption
 ADAD-ADAE Adresse de ligne après interruption
 ADAF-ADBO Adresse de la routine ON-ERROR
 ADB1 Routine de traitement des erreurs activée
 ADB2 Paramètre sound état canal
 ADB3 Paramètre sound courbe d'enveloppe du volume
 ADB4 Paramètre sound courbe d'enveloppe du ton
 ADB5-ADB6 Paramètre sound période
 ADB7 Paramètre sound période bruit
 ADB8 Paramètre sound volume
 ADB9-ADBA Paramètre sound durée
 ADBB-ADBC ENV & ENV
 ADCB-ADCF Mémoire provisoire pour nombre à virgule flottante
 ADDO-AE03 Table pour variable d'échelle
 AE04-AE05 Table pour FN
 AE06-AE0B Table pour tableaux
 AE0C-AE25 Types de variable A-Z
 AE2D Caractère de séparation pour instruction INPUT
 AE2E-AE2F Adresse de ligne pour instruction READ
 AE30-AE31 Pointeur DATA
 AE32-AE33 Mémoire pour pointeur de pile Basic
 AE34-AE35 Adresse de l'instruction actuelle
 AE36-AE37 Adresse de la ligne de programme actuelle
 AE38 Flag TRACE
 AE3F-AE40 Adresse de début pour LOAD
 AE41 Flag pour CHAIN MERGE
 AE42 Type de fichier
 AE43-AE44 Longueur de fichier
 AE45 Flag pour programme protégé
 AE46-AE78 Buffer pour conversion en ASCII
 AE72-AE73 Adresse pour instruction CALL
 AE74 Configuration pour instruction CALL
 AE75-AE76 hl pendant l'instruction CALL
 AE77-AE78 sp pendant l'instruction CALL
 AE79 Largeur du tabulateur
 AE7B-AE7C Pointeur sur HIMEM
 AE7D-AE7E Pointeur sur fin de Ram libre
 AE7F-AE80 Pointeur sur début Ram libre
 AE81-AE82 Pointeur sur début de programme

AE83-AE84 Pointeur sur fin de programme
AE85-AE86 Pointeur sur début des variables
AE87-AE88 Pointeur sur début des tableaux
AE89-AE8A Pointeur sur fin des tableaux
AE8B-B08A Pile Basic
B08B-B08C Pointeur de pile Basic
B08D-B08E Début des chaînes de caractères
B08F-B090 Fin des chaînes de caractères
B09A-B09B Pointeur sur pile de descripteur de chaîne
B09C-B0B9 Pile de descripteur de chaîne
B0BA-B0BC Descripteur de chaîne
BOC1 Type de variable
BOC2-BOC3 Variable entière ou
Adresse d'un nombre à virgule flottante
Pointeur sur descriptor de chaîne
B8E4-B8E7 Valeur RND
B8E8-B8EC Mémoire provisoire pour nombre à virgule flottante
B8ED-B8F1 Mémoire provisoire pour nombre à virgule flottante
B8F2-B8F6 Mémoire provisoire pour nombre à virgule flottante
B8F7 Flag pour DEG/RAD

3.5 Basic et langage-machine

3.5.1 L'instruction CALL

L'instruction CALL sert de lien entre le Basic et le langage-machine. Elle permet en effet d'appeler à partir d'un programme Basic un programme en langage-machine. L'instruction CALL doit être accompagnée d'une adresse 16 bits qui indique en quelle adresse figure le programme en langage-machine, par exemple:

```
CALL &8000
```

Cette instruction appellera un programme en langage-machine figurant à l'adresse &8000 ou 32768 en décimal. Si le programme en langage-machine se termine par une instruction RET, le contrôle est rendu à l'interpréteur qui poursuit l'exécution du programme Basic.

Avec l'instruction CALL on ne peut accéder directement au système d'exploitation ou à l'interpréteur Basic. Pour toute la zone d'adresses de 64 K, c'est la Ram qui est sélectionnée automatiquement. Il est cependant possible évidemment d'appeler des routines du système d'exploitation à travers les adresses d'entrée qui figurent en &B000. Ces routines s'occupent elles-mêmes de réaliser la configuration Rom/Ram qui convient. Si vous voulez accéder avec une instruction CALL à des routines de l'interpréteur Basic ou à des routines du système d'exploitation qui ne peuvent être appelées avec des vecteurs, vous pouvez utiliser les routines RST 3 et RST 5 qui réalisent la commutation.

L'instruction CALL permet cependant également de transmettre des paramètres du Basic à la routine en langage-machine. Vous pouvez pour cela transmettre jusqu'à 32 paramètres qui doivent être placés à la suite de l'instruction CALL, séparés par des virgules. Ces paramètres, ainsi que l'adresse elle-même doivent donner une valeur 16 bits. Ils sont placés par le Basic sur la pile. L'interpréteur Basic transmet l'adresse de base du bloc de paramètres dans le registre IX. Dans l'accumulateur figure le nombre de paramètres transmis. Le dernier paramètre figure donc à l'adresse IX, l'avant-dernier à l'adresse IX+2 et le premier paramètre à l'adresse IX+2*(A-1).

Pendant l'instruction CALL, les contenus de tous les registres peuvent

être modifiés. Le pointeur de pile peut lui aussi être modifié pour autant qu'on soit sûr que lors de l'exécution de l'instruction RET qui termine le programme en langage-machine, c'est bien la bonne adresse de retour qui sera retirée de la pile.

Les applications possibles de l'instruction CALL sont très diverses et vous pouvez dans ce domaine donner libre cours à votre imagination. Vous pouvez par exemple créer des fonctions graphiques nouvelles telles que le dessin de cercles, le remplissage de surfaces, etc...

La transmission de paramètres en retour, de la routine en langage-machine au Basic n'est pas prévue mais elle reste cependant possible par un petit détour. Si par exemple le résultat d'un programme en langage-machine doit être affecté à une variable, on peut transmettre l'adresse de cette variable à travers l'instruction CALL, grâce au signe 'arobas':

```
CALL &AB00,@A
```

L'adresse de la variable A sera ainsi à la disposition du programme en langage-machine qui pourra modifier directement la valeur de cette variable. Cette possibilité est décrite plus précisément dans le chapitre sur le pointeur de variable.

3.5.2 Extensions du Basic avec RSX

Le système d'exploitation et le Basic du CPC soutiennent la possibilité d'intégrer ses propres instructions dans le Basic. C'est ce qu'on appelle RSX 'Resident System eXtension'. Ces extensions peuvent être appelées en Basic à travers un nom, et elles permettent une transmission de paramètres comme nous l'avons déjà décrite pour l'instruction CALL. Si nous voulons par exemple écrire une extension graphique qui dessine un carré sur l'écran, l'appel de cette fonction se présentera ainsi:

```
10 IQADRAT,100,100,50
```

Nous voulons ainsi dessiner un carré dont l'angle supérieur gauche aura les coordonnées 100, 100 avec un côté d'une longueur de 50 points.

Comme vous voyez, une extension d'instruction est marquée par un trait

vertical (SHIFT@) placé devant le mot instruction.

Une telle extension d'instruction peut figurer dans une Rom d'extension, comme c'est le cas lorsque vous connectez le lecteur de disquette, ou bien également en Ram. Cela nous donne donc la possibilité d'écrire nos propres extensions d'instruction. Pour que le système d'exploitation sache où il doit chercher une telle extension, l'extension doit d'abord être 'intégrée'. On emploie pour cela une routine du système d'exploitation: KL LOG EXT. L'exemple suivant réalise l'instruction évoquée ci-dessus pour dessiner un carré et montre comment l'intégration se réalise.

;RSX - EXTENSIONS D'INSTRUCTION
;L.E. 21/12/84

```
BCD1      LOGEXT  EQU  &BCD1 ; intégrer extension
BBC6      ASKCUR~ EQU  &BBC6 ; aller chercher curseur graphique
BBC0      MOVABS  EQU  &BBC0 ;fixer curseur graphique
BBF9      DRAWRE  EQU  &BBF9 ;tracer ligne relativ.
BDC7      CHGSGN  EQU  &BDC7 ;modifier signe
8000      ORG     &8000
8000 010980 LD     BC,RSX ;adresse de la table
d'instructions RSX
8003 211680 LD     HL,KERNAL ;4 octets Ram pour Kernal
8006 C3D1BC JP     LOGEXT ; intégrer extension
8009 0E80    RSX  DEFW  TABLE ; Adresse des mots d'instruction
800B C31A80 JP     QUADRAT
800E 51554144 TABLE DEFM  "QUADRA"
8014 D4      DEFB  "T"+&80
8015 00      DEFB  0 ; fin de la table
8016        KERNAL DEFS  4 ; mémoire pour Kernal
801A FE03    QUADRA CP    3 ; trois paramètres?
801C C0      RET    NZ
801D CDC6BB CALL  ASKCURS ;aller chercher curseur graphique
8020 D5      PUSH  DE ; ranger coordonnée X
8021 E5      PUSH  HL ; ranger coordonnée Y
8022 DD5605 LD     D,(IX+5)
8025 DD5E04 LD     E,(IX+4) ;coordonnée X
8028 DD6603 LD     H,(IX+3)
802B DD6E02 LD     L,(IX+2) ;coordonnée Y
```

802E	CDC0BB	CALL	MOVABS ; curseur graphique sur coordonnées X, Y
8031	DD5601	LD	D,(IX+1)
8034	DD5E00	LD	E,(IX) ; longueur dans de comme offset X
8037	D5	PUSH	DE ; ranger
8038	210000	LD	HL,0 ; offset Y
803B	CDF9BB	CALL	DRAWREL ; tracer ligne horizontale
803E	E1	POP	HL
803F	E5	PUSH	HL
8040	CDC7BD	CALL	CHGSGN ; offset Y négatif
8043	E5	PUSH	HL
8044	110000	LD	DE,0
8047	CDF9BB	CALL	DRAWREL ; tracer ligne verticale
804A	D1	POP	DE ; offset X négatif
804B	210000	LD	HL,0 ; offset Y nul
804E	CDF9BB	CALL	DRAWREL ; tracer ligne horizontale
8051	E1	POP	HL
8052	110000	LD	DE,0
8055	CDF9BB	CALL	DRAWREL ; tracer ligne verticale
8058	E1	POP	HL
8059	D1	POP	DE
805A	C3C0BB	JP	MOVABS ; rétablir coordonnées

Après que ce programme ait été chargé (comme fichier binaire à partir de la cassette ou de la disquette) ou qu'il ait été placé en mémoire avec un programme de chargement de DATA, il doit être initialisé une seule fois. Il faut pour cela utiliser l'appel CALL &8000. La nouvelle instruction est alors disponible. Deux tables sont utilisées pour l'intégration. La première, appelée RSX dans notre exemple, contient tout d'abord l'adresse de la seconde table, appelée ici TABLE, suivie des instructions de saut à l'extension proprement dite. La seconde table contient les noms sous lesquels les nouvelles instructions peuvent être appelées. Les majuscules et les points sont autorisés. Le dernier caractère d'un mot instruction est marqué par son bit 7 qui est mis. La fin de la table est indiquée par un octet nul. Chaque table doit bien sûr contenir le même nombre d'entrées. Pour chaque mot d'instruction doit figurer l'adresse de saut correspondante dans la première table. Sous l'étiquette KERNAL, nous devons mettre 4 octets à la disposition du système d'exploitation qui sont utilisés pour la gestion de l'extension. Les 4 octets doivent être placés entre l'adresse &4000 et l'adresse &BFFF.

La routine de dessin d'un carré commence par l'étiquette QUADRAT (quadrate en anglais=carré). On contrôle d'abord si trois paramètres ont bien été transmis. Si ce n'est pas le cas, on quitte la routine immédiatement. Mais si c'est le cas, on va chercher la position actuelle du curseur graphique et on la range sur la pile. On va ensuite chercher dans de et hl les coordonnées X et Y transmises. La base du bloc de paramètres se trouve en IX. Après que le curseur graphique ait été fixé sur ces coordonnées, la routine de dessin d'une ligne relativement à la position actuelle peut être appelée quatre fois. Pour calculer un offset négatif, on appelle la routine CHGSGN de l'arithmétique entière. Pour finir, on rétablit la position originelle du curseur.

Voici un exemple d'utilisation de cette routine:

```
10CLS
20FOR I=35 TO 400 STEP 20
30 IQADRAT,1,1,30
40NEXT
```

3.5.3 Le pointeur de variable '@'

Une fonction particulièrement intéressante pour le programmeur en langage-machine est constituée par le pointeur de variable qui est appelé avec l'arobas. Cette fonction renvoie l'adresse où est placée une variable. L'appel de cette fonction se présente ainsi:

```
PRINT@a
```

On sort ainsi l'adresse de la variable a. Si la variable n'avait pas encore été initialisée, le message d'erreur 'Improper argument' sera sorti.

Si nous voulons maintenant accéder au contenu de la variable, nous devons distinguer entre les 3 différents types possibles.

La situation est très simple en ce qui concerne les variables entières. La valeur 16 bits est placée à l'adresse fournie. Nous pouvons donc obtenir la valeur de la variable a% avec la formule:

```
PRINT PEEK(@a%)+256*PEEK(@a%+1)
```

Nous pouvons ainsi obtenir des valeurs entre 0 et 65535. Si nous voulons tenir également compte du signe, nous devons utiliser la fonction UNT.

```
PRINT UNT(PEEK(@a%)+256*PEEK(@a%+1))
```

Pour les variables à virgule flottante, le pointeur de variable est également dirigé sur la valeur de la variable, mais celle-ci est exprimée avec 5 octets. Les 5 premiers octets sont ce qu'on appelle la mantisse et le cinquième octet est la puissance de 2 par laquelle doit être multipliée la mantisse pour obtenir la valeur de la variable. Si nous désignons les 4 octets de la mantisse par m1 à m4 et l'exposant par ex, nous obtenons la valeur à virgule flottante avec la formule suivante:

$$x=(1-2*\text{SGN}(m4 \text{ AND } 128))*2^{(ex-129)}*(1+((m4 \text{ AND } 127)+(m3+(m2+m1/256)/256)/256)/128)$$

La formule met en évidence que le signe du nombre à virgule flottante se trouve dans le bit supérieur de m4 et que les octets de la mantisse m1 à m4 ont des valeurs croissantes. La puissance de 2 contient un offset de 129 ce qui donne des valeurs de 2¹⁻¹²⁹ à 2¹¹²⁷. Essayons notre formule:

```
100a=-13:'variable a virgule flottante examinee
110ad=@a:'adresse de a
120m1=PEEK(ad):m2=PEEK(ad+1):m3=PEEK(ad+2)
130m4=PEEK(ad+3):ex=PEEK(ad+4)
140PRINT(1-2*SGN(m4 AND 128))*2^(ex-129)*
      (1+((m4 AND 127)+(m3+(m2+m1/256)/256)/256)/128)
```

Si vous faites tourner ce programme, vous obtiendrez en résultat la valeur -13. Remplacez si vous le voulez la ligne 100 par INPUT a et vous pourrez tester n'importe quelles valeurs.

La fonction de pointeur de variable trouve son application dans l'instruction CALL qui ne peut en effet transmettre que des valeurs 16 bits. Si vous voulez donc travailler avec des valeurs à virgule flottante, vous pouvez transmettre avec '@' l'adresse d'une variable à virgule flottante. Vous pourrez ensuite vous référer à cette adresse. Cette méthode permet également bien sûr de modifier directement la valeur

d'une variable à virgule flottante.

Le cas des variables alphanumériques est encore plus intéressant. Ici aussi, nous pouvons utiliser le pointeur de variable qui nous renvoie l'adresse de la variable. Ce n'est cependant pas directement l'adresse de la chaîne de caractères mais celle de ce qu'on appelle le descripteur de chaîne. Ce descripteur de chaîne est long de trois octets. Le premier octet contient la longueur de la chaîne, soit une valeur entre 0 et 255. Les deux octets suivants contiennent l'adresse de la chaîne.

```
100INPUT a$
110ad=@a$
120I=PEEK(ad)
130sa=PEEK(ad+1)+256*PEEK(ad+2)
140FOR I=sa TO sa+I-1:PRINT CHR$(PEEK(I));:NEXT
```

Ce programme va chercher la longueur et l'adresse de la chaîne, la lit et la sort.

Ici aussi, il est possible de transmettre une chaîne à l'instruction CALL à travers le pointeur de variable.

Les chaînes peuvent être encore employée en liaison avec l'instruction CALL de façon tout à fait différente. On peut par exemple placer tout simplement un programme en langage-machine dans une chaîne et l'appeler avec l'instruction CALL et le pointeur de variable. Le programme en langage-machine doit pour cela être transposable (il ne doit pas contenir d'adresse absolue interne) et il ne doit pas compter plus de 255 octets. La plupart des petits programmes utilitaires remplissent ces conditions. Si vous voulez utiliser cette méthode, il vous faut procéder ainsi:

Le programme en langage-machine est d'abord placé dans la variable alphanumérique. On utilisera le plus souvent READ et DATA à cet effet. Si vous voulez ensuite faire exécuter le programme, il vous suffit de faire calculer l'adresse de début de la chaîne de caractères (et donc du programme) avec l'arobas.

3.6 Le listing de la Rom Basic

3.6.1 L'arithmétique à virgule flottante

Toutes les fonctions arithmétiques qu'utilise l'interpréteur Basic se trouvent dans la Rom du système d'exploitation. Elles sont appelées à travers une table de saut placée en &BD3D à &BDC7. Si vous voulez modifier les routines arithmétiques, il vous suffit d'insérer à l'emplacement voulu un saut à cette routine.

Nous allons vous montrer comme exemple d'application des routines avec virgule flottante une routine de calcul de la racine carrée d'un nombre. L'interpréteur Basic du CPC nous fournit certes déjà cette fonction mais nous voulons démontrer que celle-ci peut être encore améliorée par l'emploi d'algorithmes plus puissants.

La fonction SQR intégrée travaille d'après le même algorithme que le calcul de la puissance.

$$\text{SQR}(X)=\text{EXP}(\text{LOG}(X)*0.5)$$

Il faut donc calculer chaque fois les fonction exponentielle et logarithme ce qui s'effectue à travers des calculs de polynômes compliqués et longs. La racine carré peut cependant être calculée simplement à travers un processus d'itération.

$$X(N+1)=(X(N)+A/X(N))/2$$

où A est le nombre dont la racine doit être extraite, X(N) est l'ancienne et X(N+1) la nouvelle valeur approchée. Comme valeur de départ, on peut prendre le nombre A lui-même. On obtient une meilleure valeur approchée lorsqu'on divise par deux la puissance de deux du nombre à virgule flottante. Le résultat ne se modifie plus ensuite, après 4 itérations, dans le cadre de la précision de calcul. Notez également que la division par deux n'a pas été réalisée avec une division à virgule flottante qui prend beaucoup de temps. On a simplement décrémente de 1 la puissance de deux. Le gain de temps dû à ce procédé est significatif. La routine SQR de l'interpréteur met en effet 27 millisecondes, alors que notre routine exécute la même tâche en 8 millisecondes. Elle est donc plus de trois fois plus rapide.

;ROUTINE SQR RAPIDE

;L.E. 18/12/84

AB00		ORG	&AB00
BD70	SGN	EQU	&BD70
BD64	DIV	EQU	&BD64
BD58	ADD	EQU	&BD58
AB00	CD70BD	NEWSQR	CALL SGN ;examiner signe
AB03	3F		CCF
AB04	C8		RET Z ;zéro, déjà terminé
AB05	F20CAB		JP P,GOON
AB08	3E01		LD A,1 ;'IMPROPER ARGUMENT'
AB0A	B7		OR A
AB0B	C9		RET
AB0C	E5	GOON	PUSH HL
AB0D	1153AB		LD DE,STORE1
AB10	010500		LD BC,5
AB13	EDB0		LDIR ;ranger radicande
AB15	E1		POP HL
AB16	E5		PUSH HL
AB17	DDE1		POP IX
AB19	DD7E04		LD A,(IX+4) ;puissance
AB1C	D681		SUB &81 ;normaliser
AB1E	3F		CCF
AB1F	1F		RRA ;diviser puissance par deux
AB20	C601		ADD A,1
AB22	DD7704		LD (IX+4),A ;comme valeur de départ
AB25	0604		LD B,4 ;4 itérations
AB27	C5	ITER	PUSH BC
AB28	E5		PUSH HL
AB29	1158AB		LD DE,STORE2
AB2C	010500		LD BC,5
AB2F	EDB0		LDIR ;ranger valeur approchée
AB31	E1		POP HL
AB32	E5		PUSH HL
AB33	1153AB		LD DE,STORE1
AB36	EB		EX DE,HL

AB37	010500	LD	BC,5
AB3A	EDB0	LDIR	,aller chercher radicande
AB3C	E1	POP	HL
AB3D	1158AB	LD	DE,STORE2
AB40	CD64BD	CALL	DIV
AB43	1158AB	LD	DE,STORE2
AB46	CD58BD	CALL	ADD
AB49	E5	PUSH	HL
AB4A	DDE1	POP	IX
AB4C	DD3504	DEC	(IX+4) ;nombre/2
AB4F	C1	POP	BC
AB50	10D5	DJNZ	ITER
AB52	C9	RET	
AB53	STORE1	DEFS	5
AB58	STORE2	DEFS	5

Mais comment faire pour que l'interpréteur utilise la nouvelle routine? C'est le vecteur &BD79 qui sert pour la fonction SQR. Il faut donc placer en cet endroit un saut à notre routine:

JP &AB00

Lorsque la routine est appelée en Basic, le registre HL doit être pointé sur la valeur à virgule flottante. Après exécution de la routine, le registre HL doit être pointé sur le résultat. Normalement la valeur de registre ne doit pas avoir été modifiée. Les flags indiquent l'état des erreurs de la fonction:

Etat des erreurs de la fonction:

C=1	exécution correcte
C=0 & Z=1	'Division by zero'
C=0 & N=1	'Overflow'
C=0 & Z=0	'Improper argument'

Vous trouverez dans les pages suivantes le listing de l'arithmétique à virgule flottante. Chaque routine contient également l'adresse de la table de saut à travers laquelle elle est appelée par l'interpréteur Basic. Vous trouverez ensuite au chapitre 3.6.3 l'arithmétique entière

qui est utilisée par l'interpréteur chaque fois que c'est possible. En effet comme elle ne travaille qu'avec des valeurs sur deux octets, cette arithmétique est toujours nettement plus rapide que le calcul avec des nombres à virgule flottante. Servez-vous également de ce fait dans vos programmes et utilisez autant que possible des variables entières. Cela vaut notamment pour les boucles FOR-NEXT (voyez également à ce sujet le chapitre 3.2).

ARITHMETIQUE A VIRGULE FLOTTANTE

```

***** BD3D copier variable de (de) dans (hl)
2E18 E5      push  hl
2E19 D5      push  de
2E1A C5      push  bc
2E1B EB      ex    de,hl
2E1C 010500  ld    bc,0005    copier
2E1F EDB0    ldir   5 octets
2E21 EB      ex    de,hl
2E22 2B      dec   hl
2E23 7E      ld    a,(hl)    a=exposant
2E24 C1      pop   bc
2E25 D1      pop   de
2E26 E1      pop   hl
2E27 37      scf
2E28 C9      ret

***** BD40 convertir entier en virgule flottante
2E29 D5      push  de
2E2A C5      push  bc
2E2B F67F    or    7F
2E2D 47      ld    b,a
2E2E AF      xor   a
2E2F 12      ld    (de),a
2E30 13      inc   de
2E31 12      ld    (de),a
2E32 13      inc   de
2E33 0E90    ld    c,90      exposant, 2115
2E35 7C      ld    a,h
2E36 B7      or    a
2E37 2008    jr    nz,2E41
2E39 4F      ld    c,a
2E3A 65      ld    h,l
2E3B 6F      ld    l,a
2E3C B4      or    h
2E3D 280D    jr    z,2E4C
2E3F 0E88    ld    c,88      exposant, 217
2E41 FA4B2E  jp    m,2E4B
2E44 29      add   hl,hl
2E45 0D      dec   c
2E46 B4      or    h
2E47 F2442E  jp    p,2E44
2E4A 7C      ld    a,h
2E4B A0      and   b
2E4C EB      ex    de,hl
2E4D 73      ld    (hl),e
2E4E 23      inc   hl
2E4F 77      ld    (hl),a
2E50 23      inc   hl
2E51 71      ld    (hl),c
2E52 C1      pop   bc
2E53 E1      pop   hl
2E54 C9      ret

```

ARITHMETIQUE A VIRGULE FLOTTANTE

***** BD43 convertir valeur 4 octets en virgule flott.

```

2E55 C5      push  bc
2E56 0100A0  ld      bc,A000      exposant. 2131
2E59 CD602E  call   2E60          convertir
2E5C C1      pop    bc
2E5D C9      ret
    
```

***** BD94 convertir valeur 4 octets en virgule flott.

```

2E5E 06A8    ld      b,A8          exposant, 2139
2E60 D5      push   de
2E61 CDA136  call   36A1          conversion
2E64 D1      pop    de
2E65 C9      ret
    
```

***** BD46 virgule flottante => entier

```

2E66 E5      push   hl
2E67 DDE1    pop    ix
2E69 AF      xor    a
2E6A DD9604  sub    (ix+04)       exposant
2E6D 281B    jr     z,2E8A        nombre égal zéro?
2E6F C690    add    a,90
2E71 D0      ret    nc
2E72 D5      push   de
2E73 C5      push   bc
2E74 C610    add    a,10
2E76 CD3D36  call   363D
2E79 CB21    sla    c
2E7B ED5A    adc    hl,de
2E7D 2808    jr     z,2E87
2E7F DD7E03  ld     a,(ix+03)     signe de la mantisse
2E82 B7      or     a
2E83 3F      ccf
2E84 C1      pop    bc
2E85 D1      pop    de
2E86 C9      ret

2E87 9F      sbc    a,a
2E88 18F9    jr     2E83

2E8A 6F      ld     l,a           zéro dans hl
2E8B 67      ld     h,a
2E8C 37      scf
2E8D C9      ret
    
```

***** BD49 virgule flottante => entier

```

2E8E CDA12E  call   2EA1          FIX
2E91 D0      ret    nc
2E92 F0      ret    p
2E93 E5      push   hl
2E94 79      ld     a,c
2E95 34      inc    (hl)
2E96 2006    jr     nz,2E9E
2E98 23      inc    hl
2E99 3D      dec    a
2E9A 20F9    jr     nz,2E95
    
```

ARITHMETIQUE A VIRGULE FLOTTANTE

```

2E9C 34      inc    (hl)
2E9D 0C      inc    c
2E9E E1      pop    hl
2E9F 37      scf
2EA0 C9      ret

***** BD4C FIX
2EA1 E5      push   hl
2EA2 D5      push   de
2EA3 E5      push   hl
2EA4 DDE1    pop    ix
2EA6 CD0436  call   3604      fonction FIX
2EA9 D1      pop    de
2EAA E1      pop    hl
2EAB C9      ret

***** BD4F INT
2EAC CDA12E  call   2EA1      FIX
2EAF D0      ret    nc
2EB0 C8      ret    z
2EB1 CB78    bit    7,b
2EB3 C8      ret    z
2EB4 18DD    jr     2E93

***** BD52
2EB6 CDE835  call   35E8      SGN
2EB9 47      ld     b,a
2EBA 2852    jr     z,2FOE
2EBC FCFB35  call   m,35FB    négatif,, alors inversion de signe
2EBF E5      push   hl
2EC0 DD7E04  ld     a,(ix+04)  normaliser
2EC3 D680    sub    80         exposant
2EC5 5F      ld     e,a
2EC8 9F      sbc   a,a
2EC7 57      ld     d,a
2EC8 6B      ld     l,e
2EC9 62      ld     h,d
2ECA 29      add   hl,hl
2ECB 29      add   hl,hl
2ECC 29      add   hl,hl
2ECD 19      add   hl,de
2ECE 29      add   hl,hl      fois 77
2ECF 19      add   hl,de
2ED0 29      add   hl,hl
2ED1 29      add   hl,hl
2ED2 19      add   hl,de
2ED3 7C      ld     a,h
2ED4 D609    sub    09
2ED6 5F      ld     e,a
2ED7 E1      pop    hl
2ED8 C5      push   bc
2ED9 D5      push   de
2EDA C41F2F  call   nz,2F1F    multiplier nombre par 10ja
2EDD FD21132F  ld     iy,2F13    3124999.98
2EE1 CDA035  call   35A0      comparer

```

ARITHMETIQUE A VIRGULE FLOTTANTE

2EE4	281B	jr	z,2F01	égal?
2EE6	3008	jr	nc,2EF0	supérieur?
2EE8	CD1234	call	3412	multiplication par 10
2EEB	D1	pop	de	
2EEC	1D	dec	e	
2EED	D5	push	de	
2EEE	18ED	jr	2EDD	

2EF0	FD21182F	ld	iy,2F18	1E9
2EF4	CDA035	call	35A0	comparer
2EF7	3808	jr	c,2F01	inférieur?
2EF9	CD9B34	call	349B	division par 10
2EFC	D1	pop	de	
2EFD	1C	inc	e	
2EFE	D5	push	de	
2EFF	18EF	jr	2EFO	

2F01	CD8E2E	call	2E8E
2F04	79	ld	a,c
2F05	D1	pop	de
2F06	C1	pop	bc
2F07	4F	ld	c,a
2F08	3D	dec	a
2F09	85	add	a,l
2F0A	6F	ld	l,a
2F0B	D0	ret	nc
2F0C	24	inc	h
2F0D	C9	ret	

2F0E	5F	ld	e,a
2F0F	77	ld	(hl),a
2F10	0E01	ld	c,01
2F12	C9	ret	

2F13	F0 1F BC 3E 96	3124999.98
2F18	FE 27 6B 6E 9E	1E9

***** BD55 multiplier nombre par 10↑a

2F1D	2F	cpl	a
2F1E	3C	inc	a
2F1F	B7	or	a
2F20	37	scf	
2F21	C8	ret	z
2F22	4F	ld	c,a
2F23	F2282F	jp	p,2F28
2F26	2F	cpl	a
2F27	3C	inc	a
2F28	CD3E2F	call	2F3E
2F2B	2809	jr	z,2F36
2F2D	C5	push	bc
2F2E	F5	push	af
2F2F	CD362F	call	2F36
2F32	F1	pop	af
2F33	C1	pop	bc

ARITHMETIQUE A VIRGULE FLOTTANTE

```

2F34 18F2      jr      2F28

2F36 79        ld      a,c
2F37 87        or      a
2F38 F29E34    jp      p,349E      division
2F3B C31534    jp      3415        multiplication

2F3E 118F2F    ld      de,2F8F     1E13
2F41 D60D      sub     0D          -13
2F43 D0        ret     nc          supérieur égal?
2F44 C60C      add     a,0C       +12
2F46 5F        ld      e,a
2F47 87        add     a,a
2F48 87        add     a,a         fois 5
2F49 83        add     a,e
2F4A C653      add     a,53
2F4C 5F        ld      e,a         2F53, puissances de 10
2F4D CE2F      adc     a,2F
2F4F 93        sub     e
2F50 57        ld      d,a
2F51 AF        xor     a
2F52 C9        ret
    
```

*****.constantes à virgule flottante

```

2F53 00 00 00 20 84      10
2F58 00 00 00 48 87      100
2F5D 00 00 00 7A 8A      1000
2F62 00 00 40 1C 8E      10000
2F67 00 00 50 43 91      100000
2F6C 00 00 24 74 94      1000000
2F71 00 80 96 18 98      10000000
2F76 00 20 BC 3E 9B      100000000
2F7B 00 28 6B 6E 9E      1E9
2F80 00 F9 02 15 A2      1E10
2F85 40 B7 43 3A A5      1E11
2F8A 10 B5 D4 68 A8      1E12
2F8F 2A E7 84 11 AC      1E13
    
```

***** BD97 RND Init

```

2F94 216589      ld      hl,8965
2F97 22E6B8      ld      (B8E6),hl
2F9A 21076C      ld      hl,6C07
2F9D 22E4B8      ld      (B8E4),hl
2FA0 C9          ret
    
```

***** BD9A Random Seed

```

2FA1 EB          ex      de,hl
2FA2 CD942F      call   2F94         RND Init
2FA5 EB          ex      de,hl
2FA6 CDE835      call   35E8         SGN
2FA9 C8          ret     z
2FAA 11E4B8      ld      de,B8E4     pointeur sur mantisse RND
2FAD 0604        ld      b,04        4 octets
2FAF 1A          ld      a,(de)
2FB0 AE          xor     (hl)         créer nouvelle mantisse
    
```


ARITHMETIQUE A VIRGULE FLOTTANTE

```

2FB1 12      ld  (de),a
2FB2 13      inc de
2FB3 23      inc hl
2FB4 10F9    djnz 2FAF      octet suivant
2FB6 C9      ret
    
```

***** BD9D RND

```

2FB7 E5      push hl
2FB8 2AE6B8  ld  hl,(B8E6)
2FB9 01076C  ld  bc,6C07
2FBA CDFAC2F call 2FFA
2FC1 E5      push hl
2FC2 2AE4B8  ld  hl,(B8E4)
2FC3 016589  ld  bc,8965
2FC4 CDFAC2F call 2FFA
2FC5 D5      push de
2FC6 E5      push hl
2FC7 2AE6B8  ld  hl,(B8E6)
2FC8 CDFAC2F call 2FFA
2FD0 E3      ex  (sp),hl
2FD1 09      add hl,hl
2FD2 22E4B8  ld  hl,(B8E4),hl
2FD3 E1      pop  hl
2FD4 01076C  ld  bc,6C07
2FD5 ED4A    adc hl,hl
2FD6 C1      pop  bc
2FD7 09      add hl,hl
2FD8 C1      pop  bc
2FD9 09      add hl,hl
2FE0 C1      pop  bc
2FE1 09      add hl,hl
2FE2 22E6B8  ld  hl,(B8E6),hl
2FE3 E1      pop  hl
    
```

***** BDA0 amener dernière valeur RND

```

2FE6 E5      push hl
2FE7 DDE1    pop  ix
2FE8 2AE4B8  ld  hl,(B8E4)
2FE9 ED5BE6B8 ld  de,(B8E6)
2FF0 010000  ld  bc,0000
2FF1 DD360480 ld  (ix+04),80      exposant
2FF2 C3B136  jp  36B1
    
```

```

2FFA EB      ex  de,hl
2FFB 210000  ld  hl,0000
2FFE 3E11    ld  a,11
3000 3D      dec a
3001 C8      ret  z
3002 29      add hl,hl
3003 CB13    rl  e
3004 CB12    rl  d
3005 30F7    jr  nc,3000
3006 09      add hl,hl
3007 30F4    jr  nc,3000
3008 13      inc de
3009 18F1    jr  3000
    
```

ARITHMETIQUE A VIRGULE FLOTTANTE

```

***** BD82 LOG10
300F 118B30      ld   de,308B      LOG10(2)
3012 1803        jr    3017

```

```

***** BD7F LOG
3014 118630      ld   de,3086      LOG(2)
3017 CDE835      call 35E8        SGN
301A 3D          dec  a
301B FE01        cp   01
301D D0          ret  nc
301E D5          push de
301F CD6C35      call 356C        tester exposant
3022 F5          push af
3023 DD360480    ld   (ix+04),80  exposant, nombre 0.5 à 1
3027 118130      ld   de,3081      1/SQR(2)
302A CD9A35      call 359A        comparer
302D 3006        jr    nc,3035     supérieur?
302F DD3404      inc (ix+04)      augmenter exposant, nombre doublé
3032 F1          pop  af
3033 3D          dec  a
3034 F5          push af
3035 CD1633      call 3316        stockage provisoire du résultat
3038 D5          push de
3039 113233      ld   de,3332      1
303C CD3F33      call 333F        Addition
303F EB          ex   de,hl
3040 E1          pop  hl
3041 D5          push de
3042 113233      ld   de,3332      1
3045 CD3733      call 3337        soustraction
3048 D1          pop  de
3049 CD9E34      call 349E        division
304C CDA932      call 32A9        calcul de polynôme

```

```

***** constantes à virgule flottante pour LOG
304F 04          degré de polynôme
3050 4C 4B 57 5E 7F 0.434259751
3055 0D 08 9B 13 80 0.576584342
305A 23 93 38 76 80 0.961800762
305F 20 3B AA 38 82 2.88539007

```

```

*****
3064 D5          push de
3065 CD1534      call 3415        multiplication
3068 D1          pop  de
3069 E3          ex   (sp),hl
306A 7C          ld   a,h
306B B7          or   a
306C F27130      jp   p,3071
306F 2F          cpl a
3070 3C          inc  a
3071 6F          ld   l,a
3072 7C          ld   a,h
3073 2600        ld   h,00
3075 CD292E      call 2E29        convertir entier en virgule flottante

```

ARITHMETIQUE A VIRGULE FLOTTANTE

```

3078 EB          ex   de,hl
3079 E1          pop  hl
307A CD3F33     call 333F      Addition
307D D1          pop  de
307E C31534     jp   3415      multiplication

*****
3081 34 F3 04 35 80      .707106781 1/SQR(2)
3086 F8 17 72 31 80      .693147181 LOG(2)
308B 85 9A 20 1A 7F      .301029996 LOG(0(2)

***** BD85 EXP
3090 06E1        ld   b,E1
3092 CD0733     call 3307      comparer exposant
3095 D22833     jp   nc,3328    l comme résultat
3098 110031     ld   de,3100    LOG(plus grand nombre représentable)
309B CD9A35     call 359A      comparer
309E F2EC36     jp   p,36EC     supérieur, alors dépassement
30A1 110531     ld   de,3105    LOG(plus petit nombre représentable)
30A4 CD9A35     call 359A      comparer
30A7 FAE636     jp   m,36E6     inférieur, alors dépassement par le bas,
30AA 11FB30     ld   de,30FB    1/LOG(2)
30AD CDD432     call 32D4      zéro
30B0 7B        ld   a,e
30B1 F2B630     jp   p,30B6
30B4 ED44       neg  a
30B6 F5        push af
30B7 CD1D33     call 331D      multiplier
30BA CDF33     call 330F      stockage provisoire variable
30BD D5        push de
30BE CDAC32     call 32AC      calcul de polynôme

***** constantes à virgule flottante pour EXP
30C1 03          degré de polynôme
30C2 F4 32 EB 0F 73      6.86258E-5
30C7 08 B8 D5 52 7B      2.57367E-2
30CC 00 00 00 00 80      0.5

*****
30D1 E3          ex   (sp),hl
30D2 CDAC32     call 32AC      calcul de polynôme

***** constantes à virgule flottante pour EXP
30D5 02          degré de polynôme
30D6 09 60 DE 01 78      1.98164E-3
30DB F8 17 72 31 7E      0.173286795

*****
30E0 CD1534     call 3415      multiplication
30E3 D1          pop  de
30E4 E5          push hl
30E5 EB          ex   de,hl
30E6 CD3733     call 3337      soustraction
30E9 EB          ex   de,hl
30EA E1          pop  hl
30EB CD9E34     call 349E      Division

```

ARITHMETIQUE A VIRGULE FLOTTANTE

```

30EE 11CC30      ld  de,30CC      0.5
30F1 CD3F33      call 333F      Addition
30F4 DD3404      inc  (ix+04)    augmenter exposant, nombre doublé
30F7 F1          pop  af
30F8 C37B35      jp   357B      multiplier nombre par 2↑A

*****
30FB 29 3B AA 38 81      1.44269504 1/LOG(2)
3100 C7 33 0F 30 87      88.0269919 LOG(plus grand nombre)
3105 F8 17 72 B1 87      -88.7228391 LOG(plus petit nombre)
*****
310A 11CC30      ld  de,30CC      0.5
*****
***** BD7C élévation à la puissance
310D EB          ex  de,hl
310E CDE835      call 35E8      SGN, signe de l'exposant
3111 EB          ex  de,hl
3112 CA2833      jp   z,3328    zéro, alors 1 comme résultat
3115 F5          push af
3116 CDE835      call 35E8      SGN, signe de la base
3119 2825      jr   z,3140
311B 47          ld  b,a
311C FCFB35      call m,35FB    négatif, alors changer signe
311F E5          push hl
3120 CD8231      call 3182
3123 E1          pop  hl
3124 3825      jr   c,314B
3126 E3          ex  (sp),hl
3127 E1          pop  hl
3128 FA4831      jp   m,3148
312B C5          push bc
312C D5          push de
312D CD1430      call 3014      LOG
3130 D1          pop  de
3131 DC1534      call c,3415    multiplication
3134 DC9030      call c,3090    EXP
3137 C1          pop  bc
3138 D0          ret  nc
3139 78          ld  a,b
313A B7          or  a
313B FCFB35      call m,35FB
313E 37          scf
313F C9          ret

3140 F1          pop  af
3141 37          scf
3142 F0          ret  p
3143 CDEC36      call 36EC      dépassement
3146 AF          xor  a
3147 C9          ret

3148 AF          xor  a
3149 3C          inc  a
314A C9          ret

```

ARITHMETIQUE A VIRGULE FLOTTANTE

314B	4F	ld	c,a	
314C	F1	pop	af	
314D	C5	push	bc	
314E	F5	push	af	
314F	79	ld	a,c	
3150	37	scf		
3151	8F	adc	a,a	
3152	30FD	jr	nc,3151	
3154	47	ld	b,a	
3155	CD0F33	call	330F	stockage provisoire variable
3158	EB	ex	de,hl	
3159	78	ld	a,b	
315A	87	add	a,a	
315B	2815	jr	z,3172	
315D	F5	push	af	
315E	CD1D33	call	331D	multiplier par résultat intermédiaire
3161	3016	jr	nc,3179	
3163	F1	pop	af	
3164	30F4	jr	nc,315A	
3166	F5	push	af	
3167	11E8B8	ld	de,B8E8	
316A	CD1534	call	3415	multiplication
316D	300A	jr	nc,3179	
316F	F1	pop	af	
3170	18E8	jr	315A	
3172	F1	pop	af	
3173	37	scf		
3174	FCFD32	call	m,32FD	former complément
3177	18BE	jr	3137	
3179	F1	pop	af	
317A	F1	pop	af	
317B	C1	pop	bc	
317C	FAE636	jp	m,36E6	dépassement par le bas, zéro
317F	C3EE36	jp	36EE	dépassement
3182	C5	push	bc	
3183	CD1733	call	3317	aller chercher résultat intermédiaire
3186	CDA12E	call	2EA1	FIX
3189	79	ld	a,c	
318A	C1	pop	bc	
318B	3002	jr	nc,318F	
318D	2803	jr	z,3192	
318F	78	ld	a,b	
3190	B7	or	a	
3191	C9	ret		
3192	4F	ld	c,a	
3193	7E	ld	a,(hl)	
3194	1F	rra		
3195	9F	sbc	a,a	
3196	A0	and	b	
3197	47	ld	b,a	
3198	79	ld	a,c	

ARITHMETIQUE A VIRGULE FLOTTANTE

```

3199 FE02      cp    02
319B 9F       sbc   a,a
319C D0       ret   nc
319D 7E       ld    a,(hl)
319E FE27     cp    27
31A0 D8       ret   c
31A1 AF       xor   a
31A2 C9       ret
    
```

```

***** BD76 PI
31A3 11A931   ld    de,31A9      r
31A6 C3182E   jp    2E18         aller chercher variable
    
```

```

*****
31A9 A2 DA OF 49 82      3.14159265 r
    
```

```

***** BD73 DEG/RAD
31AE 32F7B8   ld    (B8F7),a
31B1 C9       ret
    
```

```

***** BD8B COS
31B2 CDE835   call  35E8        SGN
31B5 FCFB35   call  m,35FB     négatif, alors changer signe
31B8 F601     or    01
31BA 1801     jr    31BD
    
```

```

***** BD88 SIN
31BC AF       xor   a
31BD F5       push  af
31BE 111D32   ld    de,321D    1/r
31C1 06F0     ld    b,F0
31C3 3AF7B8   ld    a,(B8F7)   DEG ?
31C6 B7       or    a
31C7 2805     jr    z,31CE
31C9 112232   ld    de,3222    1/180
31CC 06F6     ld    b,F6
31CE CD0733   call  3307        comparer exposant
31D1 303A     jr    nc,320D
31D3 F1       pop   af
31D4 CDD532   call  32D5
31D7 D0       ret   nc
31D8 7B       ld    a,e
31D9 1F       rra
31DA DCFB35   call  c,35FB     changer signe
31DD 06E8     ld    b,E8
31DF CD0733   call  3307        comparer exposant
31E2 D2E636   jp    nc,36E6    dépassement par le bas, zéro
31E5 DD3404   inc  (ix+04)     augmenter exposant, nombre doublé
31E8 CDA932   call  32A9        calcul de polynôme
    
```

```

***** constantes à virgule flottante pour SIN
31EB 06      degré de polynôme
31EC 1B 2D 1A E6 6E      -3.42879E-6
31F1 F8 FB 07 28 74      1.60247E-4
31F6 01 89 68 99 79      -4.68165E-3
    
```


ARITHMETIQUE A VIRGULE FLOTTANTE

328A FE FF FF FF 7F

0.5

```
*****
328F CD1534      call 3415      multiplication
3292 F1          pop  af
3293 110532      ld  de,3205    π/2
3296 F43B33      call p,333B    soustraction
3299 3AF7B8      ld  a,(B8F7)   DEG?
329C B7          or   a
329D 112C32      ld  de,322C    180/π
32A0 C41534      call nz,3415   si DEG, alors multiplier
32A3 F1          pop  af
32A4 FCFB35      call m,35FB    négatif, alors changer signe
32A7 37          scf
32A8 C9          ret
```

```
***** calcul de polynôme
32A9 CD1D33      call 331D      multiplier
32AC CD1633      call 3316      stockage provisoire variable
32AF EB          ex  de,hl
32B0 D1          pop  de
32B1 1A          ld  a,(de)     aller chercher degré polynôme
32B2 13          inc de
32B3 47          ld  b,a       dans b
32B4 CD182E      call 2E18      aller chercher variable
32B7 13          inc de
32B8 13          inc de
32B9 13          inc de        plus 5, prochain coefficient
32BA 13          inc de
32BB 13          inc de
32BC D5          push de
32BD 11EDB8      ld  de,B8ED    stockage provisoire
32C0 05          dec  b        prochain coefficient
32C1 C8          ret  z
32C2 C5          push bc
32C3 11F2B8      ld  de,B8F2    stockage provisoire
32C6 CD1534      call 3415      multiplication
32C9 C1          pop  bc
32CA D1          pop  de
32CB D5          push de
32CC C5          push bc
32CD CD3F33      call 333F      Addition
32D0 C1          pop  bc
32D1 D1          pop  de
32D2 18E3        jr   32B7
```

```
*****
32D4 AF          xor  a
32D5 F5          push af
32D6 CD1534      call 3415      multiplication
32D9 F1          pop  af
32DA 11CC30      ld  de,30CC    0.5
32DD C43F33      call nz,333F   Addition
32E0 E5          push hl
32E1 CD662E      call 2E66      virgule flottante à entier
```


ARITHMETIQUE A VIRGULE FLOTTANTE

```

32E4 3013      jr    nc,32F9
32E6 D1        pop   de
32E7 E5        push  hl
32E8 F5        push  af
32E9 D5        push  de
32EA 11EDB8    ld    de,B8ED
32ED CD292E    call  2E29      convertir entier en virgule flottante
32F0 EB        ex    de,hl
32F1 E1        pop   hl
32F2 CD3733    call  3337      soustraction
32F5 F1        pop   af
32F6 D1        pop   de
32F7 37        scf
32F8 C9        ret

32F9 E1        pop   hl
32FA AF        xor   a
32FB 3C        inc  a
32FC C9        ret

```

*****former complément

```

32FD CD1633    call  3316      stockage provisoire de variable
3300 EB        ex    de,hl
3301 CD2833    call  3328      aller chercher 1
3304 C39E34    jp    349E      Division

```

*****comparer exposant

```

3307 CD6C35    call  356C
330A F0        ret   p
330B B8        cp   b
330C C8        ret   z
330D 3F        ccf
330E C9        ret

```

*****stockage provisoire de variable

```

330F EB        ex    de,hl
3310 21E8B8    ld    hl,B8E8  adresse objet
3313 C3182E    jp    2E18      copier variable

```

*****stockage provisoire de variable

```

3316 EB        ex    de,hl
3317 21F2B8    ld    hl,B8F2
331A C3182E    jp    2E18      aller chercher variable

```

```

331D EB        ex    de,hl
331E 21EDB8    ld    hl,B8ED
3321 CD182E    call  2E18      aller chercher variable
3324 EB        ex    de,hl
3325 C31534    jp    3415      multiplication

```

***** aller chercher constante 1

```

3328 D5        push  de
3329 113233    ld    de,3332  1
332C CD182E    call  2E18      aller chercher variable

```

ARITHMETIQUE A VIRGULE FLOTTANTE

```

332F D1      pop  de
3330 37      scf
3331 C9      ret

*****
3332 00 00 00 00 B1          1

*****BD5B soustraction (hl):=(hl)-(de)
3337 3E01      ld  a,01
3339 1805      jr  3340

*****BD5E soustraction (hl):=(hl)-(de)
333B 3E80      ld  a,80
333D 1801      jr  3340

*****BD58 Addition (hl):=(hl)+(de)
333F AF        xor  a      annuler carry
3340 E5        push hl
3341 DDE1      pop  ix
3343 D5        push de
3344 FDE1      pop  iy
3346 DD4803    ld  b,(ix+03)  signe premier operande
3349 FD4E03    ld  c,(iy+03)  signe second operande
334C B7        or   a
334D 280B      jr  z,335A
334F FA5833    jp  m,3358
3352 3E80      ld  a,80
3354 A9        xor  c
3355 4F        ld  c,a
3356 1802      jr  335A

3358 A8        xor  b
3359 47        ld  b,a
335A DD7E04    ld  a,(ix+04)  exposants
335D FDBE04    cp  (iy+04)    comparer
3360 3014      jr  nc,3376
3362 50        ld  d,b
3363 41        ld  b,c
3364 4A        ld  c,d
3365 B7        or   a
3366 57        ld  d,a
3367 FD7E04    ld  a,(iy+04)  exposant
336A DD7704    ld  (ix+04),a  exposant
336D 2854      jr  z,33C3
336F 92        sub d
3370 FE21      cp  21
3372 304F      jr  nc,33C3
3374 1811      jr  3387

3376 AF        xor  a
3377 FD9604    sub (iy+04)    exposant
337A 2859      jr  z,33D5
337C DD8604    add a,(ix+04)  exposant
337F FE21      cp  21
3381 3052      jr  nc,33D5

```

ARITHMETIQUE A VIRGULE FLOTTANTE

3383	E5	push	hl	
3384	FDE1	pop	iy	
3386	EB	ex	de,hl	
3387	5F	ld	e,a	
3388	78	ld	a,b	
3389	A9	xor	c	
338A	F5	push	af	
338B	C5	push	bc	
338C	7B	ld	a,e	
338D	CD4336	call	3643	
3390	79	ld	a,c	
3391	C1	pop	bc	
3392	4F	ld	c,a	
3393	F1	pop	af	
3394	FADA33	jp	m,33DA	
3397	FD7E00	ld	a,(iy+00)	
339A	85	add	a,l	
339B	6F	ld	l,a	
339C	FD7E01	ld	a,(iy+01)	
339F	8C	adc	a,h	
33A0	67	ld	h,a	
33A1	FD7E02	ld	a,(iy+02)	
33A4	8B	adc	a,e	
33A5	5F	ld	e,a	
33A6	FD7E03	ld	a,(iy+03)	
33A9	CBFF	set	7,a	
33AB	8A	adc	a,d	
33AC	57	ld	d,a	
33AD	D2BA36	jp	nc,36BA	
33B0	CB1A	rr	d	
33B2	CB1B	rr	e	
33B4	CB1C	rr	h	
33B6	CB1D	rr	l	
33B8	CB19	rr	c	
33BA	DD3404	inc	(ix+04)	augmenter exposant
33BD	C2BA36	jp	nz,36BA	
33C0	C3EE36	jp	36EE	dépassement

33C3	FD7E02	ld	a,(iy+02)	
33C6	DD7702	ld	(ix+02),a	
33C9	FD7E01	ld	a,(iy+01)	
33CC	DD7701	ld	(ix+01),a	
33CF	FD7E00	ld	a,(iy+00)	
33D2	DD7700	ld	(ix+00),a	
33D5	DD7003	ld	(ix+03),b	
33D8	37	scf		
33D9	C9	ret		
33DA	AF	xor	a	
33DB	91	sub	c	
33DC	4F	ld	c,a	
33DD	FD7E00	ld	a,(iy+00)	
33E0	9D	sbc	a,j	
33E1	6F	ld	l,a	

ARITHMETIQUE A VIRGULE FLOTTANTE

```

33E2 FD7E01      ld  a,(iy+01)
33E5 9C          sbc  a,h
33E6 67          ld  h,a
33E7 FD7E02      ld  a,(iy+02)
33EA 9B          sbc  a,e
33EB 5F          ld  e,a
33EC FD7E03      ld  a,(iy+03)
33EF CBFF        set  7,a
33F1 9A          sbc  a,d
33F2 57          ld  d,a
33F3 3016        jr   nc,340B
33F5 78          ld  a,b
33F6 2F          cpl  a
33F7 47          ld  b,a
33F8 AF          xor  a
33F9 91          sub  c
33FA 4F          ld  c,a
33FB 3E00        ld  a,00
33FD 9D          sbc  a,l
33FE 6F          ld  l,a
33FF 3E00        ld  a,00
3401 9C          sbc  a,h
3402 67          ld  h,a
3403 3E00        ld  a,00
3405 9B          sbc  a,e
3406 5F          ld  e,a
3407 3E00        ld  a,00
3409 9A          sbc  a,d
340A 57          ld  d,a
340B 87          add  a,a
340C DABA36      jp   c,36BA
340F C3B136      jp   36B1

```

```

***** multiplication par
3412 11532F      ld  de,2F53      10

```

```

*****BD61 multiplication
3415 D5          push de
3416 FDE1        pop  iy
3418 E5          push hl
3419 DDE1        pop  ix
341B FD7E04      ld  a,(iy+04)    exposant
341E B7          or   a
341F 282C        jr   z,344D
3421 3D          dec  a
3422 CD4835      call 3548
3425 2826        jr   z,344D
3427 3021        jr   nc,344A
3429 F5          push af
342A C5          push bc
342B CD5034      call 3450
342E 79          ld  a,c
342F C1          pop  bc
3430 4F          ld  c,a
3431 F1          pop  af

```

ARITHMETIQUE A VIRGULE FLOTTANTE

3432	CB7A	bit	7,d	
3434	200D	jr	nz,3443	
3436	3D	dec	a	
3437	2814	jr	z,344D	
3439	CB21	sla	c	
343B	CB15	rl	l	
343D	CB14	rl	h	
343F	CB13	rl	e	
3441	CB12	rl	d	
3443	DD7704	ld	(ix+04),a	exposant
3446	B7	or	a	
3447	C2BA36	jp	nz,36BA	
344A	C3EE36	jp	36EE	dépassement
344D	C3E636	jp	36E6	dépassement par le bas

3450	210000	ld	hl,0000	
3453	5D	ld	e,l	
3454	54	ld	d,h	
3455	FD7E00	ld	a,(iy+00)	
3458	CD9334	call	3493	
345B	FD7E01	ld	a,(iy+01)	
345E	CD9334	call	3493	
3461	FD7E02	ld	a,(iy+02)	
3464	CD9334	call	3493	
3467	FD7E03	ld	a,(iy+03)	
346A	F680	or	80	
346C	0608	ld	b,08	
346E	1F	rra		
346F	4F	ld	c,a	
3470	3014	jr	nc,3486	
3472	7D	ld	a,l	
3473	DD8600	add	a,(ix+00)	
3476	6F	ld	l,a	
3477	7C	ld	a,h	
3478	DD8E01	adc	a,(ix+01)	
347B	67	ld	h,a	
347C	7B	ld	a,e	
347D	DD8E02	adc	a,(ix+02)	
3480	5F	ld	e,a	
3481	7A	ld	a,d	
3482	DD8E03	adc	a,(ix+03)	
3485	57	ld	d,a	
3486	CB1A	rr	d	
3488	CB1B	rr	e	
348A	CB1C	rr	h	
348C	CB1D	rr	l	
348E	CB19	rr	c	
3490	10DE	djnz	3470	
3492	C9	ret		

ARITHMETIQUE A VIRGULE FLOTTANTE

```

3493 B7      or      a
3494 20D6   jr      nz,346C
3496 6C      ld      l,h
3497 63      ld      h,e
3498 5A      ld      e,d
3499 57      ld      d,a
349A C9      ret
    
```

```

***** division par 10
349B 11532F ld      de,2F53
    
```

```

***** BD64 division
    
```

```

349E D5      push   de
349F FDE1    pop    iy
34A1 E5      push   hl
34A2 DDE1    pop    ix
34A4 AF      xor    a
34A5 FD9604 sub    (iy+04)      exposant
34A8 2858   jr      z,3502
34AA CD4835 call   3548
34AD CAE636 jp     z,36E6      dépassement par le bas
34B0 304D   jr      nc,34FF
34B2 C5      push   bc
34B3 4F      ld     c,a
34B4 5E      ld     e,(hl)
34B5 23     inc   hl
34B6 56     ld     d,(hl)
34B7 23     inc   hl
34B8 7E     ld     a,(hl)
34B9 23     inc   hl
34BA 66     ld     h,(hl)
34BB 6F     ld     l,a
34BC EB      ex     de,hl
34BD FD4603 ld     b,(iy+03)
34C0 CBF8   set    7,b
34C2 CD3235 call   3532
34C5 3006   jr      nc,34CD
34C7 79     ld     a,c
34C8 B7      or     a
34C9 2008   jr      nz,34D3
34CB 1831   jr      34FE

34CD 0D      dec    c
34CE 29     add   hl,hl
34CF CB13    rl     e
34D1 CB12    rl     d
34D3 DD7104 ld     (ix+04),c      exposant
34D6 CD0735 call   3507
34D9 DD7103 ld     (ix+03),c
34DC CD0735 call   3507
34DF BD7102 ld     (ix+02),c
34E2 CD0735 call   3507
34E5 DD7101 ld     (ix+01),c
34E8 CD0735 call   3507
34EB D43235 call   nc,3532
    
```

ARITHMETIQUE A VIRGULE FLOTTANTE

34EE	9F	sbc	a,a	
34EF	69	ld	l,c	
34F0	DD6601	ld	h,(ix+01)	
34F3	DD5E02	ld	e,(ix+02)	
34F6	DD5603	ld	d,(ix+03)	
34F9	C1	pop	bc	
34FA	4F	ld	c,a	
34FB	C3BA36	jp	36BA	
34FE	C1	pop	bc	
34FF	C3EE36	jp	36EE	dépassement
3502	CD9435	call	3594	
3505	AF	xor	a	
3506	C9	ret		
3507	0E01	ld	c,01	
3509	3808	jr	c,3513	
350B	7A	ld	a,d	
350C	B8	cp	b	
350D	3F	ccf		
350E	CC3635	call	z,3536	
3511	3013	jr	nc,3528	
3513	7D	ld	a,l	
3514	FD9600	sub	(iy+00)	
3517	6F	ld	l,a	
3518	7C	ld	a,h	
3519	FD9E01	sbc	a,(iy+01)	
351C	67	ld	h,a	
351D	7B	ld	a,e	
351E	FD9E02	sbc	a,(iy+02)	
3521	5F	ld	e,a	
3522	7A	ld	a,d	
3523	98	sbc	a,b	
3524	57	ld	d,a	
3525	37	scf		
3526	CB11	rl	c	
3528	9F	sbc	a,a	
3529	29	add	hl,hl	
352A	CB13	rl	e	
352C	CB12	rl	d	
352E	3C	inc	a	
352F	20D8	jr	nz,3509	
3531	C9	ret		
3532	7A	ld	a,d	
3533	B8	cp	b	
3534	3F	ccf		
3535	C0	ret	nz	
3536	7B	ld	a,e	
3537	FDBE02	cp	(iy+02)	
353A	3F	ccf		
353B	C0	ret	nz	
353C	7C	ld	a,h	
353D	FDBE01	cp	(iy+01)	

ARITHMETIQUE A VIRGULE FLOTTANTE

3540	3F	ccf		
3541	C0	ret	nz	
3542	7D	ld	a,j	
3543	FDBE00	cp	(iy+00)	
3546	3F	ccf		
3547	C9	ret		
3548	4F	ld	c,a	
3549	DD7E03	ld	a,(ix+03)	
354C	FDAE03	xor	(iy+03)	
354F	47	ld	b,a	
3550	DD7E04	ld	a,(ix+04)	exposant
3553	B7	or	a	
3554	C8	ret	z	
3555	81	add	.a,c	
3556	4F	ld	c,a	
3557	1F	rra		
3558	A9	xor	c	
3559	79	ld	a,c	
355A	F26835	jp	p,3568	
355D	DDCB03FE	set	7,(ix+03)	signe négatif
3561	D67F	sub	7F	
3563	37	scf		
3564	C0	ret	nz	
3565	FE01	cp	01	
3567	C9	ret		
3568	B7	or	a	
3569	F8	ret	m	
356A	AF	xor	a	
356B	C9	ret		
356C	E5	push	hl	
356D	DDE1	pop	ix	
356F	DD7E04	ld	a,(ix+04)	exposant
3572	B7	or	a	
3573	C8	ret	z	
3574	D680	sub	80	
3576	37	scf		
3577	C9	ret		
***** BD67 multiplier par 2^a				
3578	E5	push	hl	
3579	DDE1	pop	ix	
357B	B7	or	a	puissance de deux dans accu
357C	FA8935	jp	m,3589	négatif?
357F	DD8604	add	a,(ix+04)	augmenter exposant
3582	DD7704	ld	(ix+04),a	et sauvegarder à nouveau
3585	3F	ccf		
3586	D8	ret	c	
3587	180B	jr	3594	dépassement?

ARITHMETIQUE A VIRGULE FLOTTANTE

3589	DD8604	add	a,(ix+04)	additionner exposant
358C	3802	jr	c,3590	pas de dépassement par le bas
358E	AF	xor	a	zéro comme résultat
358F	37	scf		
3590	DD7704	ld	(ix+04),a	sauvegarder à nouveau l'exposant
3593	C9	ret		

3594	DD4603	ld	b,(ix+03)	signe de la mantisse
3597	CDEE36	call	36EE	dépassement

***** BD6A comparer

359A	E5	push	hl	
359B	DDE1	pop	ix	
359D	D5	push	de	
359E	FDE1	pop	iy	
35A0	DD7E04	ld	a,(ix+04)	comparer
35A3	FDBE04	cp	(iy+04)	exposants
35A6	383A	jr	c,35E2	
35A8	2033	jr	nz,35DD	
35AA	B7	or	a	
35AB	C8	ret	z	
35AC	DD7E03	ld	a,(ix+03)	
35AF	FDAE03	xor	(iy+03)	
35B2	FADD35	jp	m,35DD	
35B5	DD7E03	ld	a,(ix+03)	
35B8	FD9603	sub	(iy+03)	
35BB	2017	jr	nz,35D4	
35BD	DD7E02	ld	a,(ix+02)	
35C0	FD9602	sub	(iy+02)	
35C3	200F	jr	nz,35D4	
35C5	DD7E01	ld	a,(ix+01)	
35C8	FD9601	sub	(iy+01)	
35CB	2007	jr	nz,35D4	
35CD	DD7E00	ld	a,(ix+00)	
35D0	FD9600	sub	(iy+00)	
35D3	C8	ret	z	
35D4	9F	sbc	a,a	
35D5	FDAE03	xor	(iy+03)	
35D8	87	add	a,a	
35D9	9F	sbc	a,a	
35DA	D8	ret	c	
35DB	3C	inc	a	
35DC	C9	ret		
35DD	DD7E03	ld	a,(ix+03)	
35E0	18F6	jr	35D8	
35E2	FD7E03	ld	a,(iy+03)	
35E5	2F	cpl	a	
35E6	18F0	jr	35D8	

***** BD70 SGN

35E8	E5	push	hl	
35E9	DDE1	pop	ix	
35EB	DD7E04	ld	a,(ix+04)	exposant

ARITHMETIQUE A VIRGULE FLOTTANTE

```

35EE B7      or  a
35EF C8      ret  z
35F0 DD7E03 ld  a,(ix+03)
35F3 87      add a,a
35F4 9F      sbc a,a
35F5 D8      ret  c
35F6 3C      inc a
35F7 C9      ret

```

***** BD6D changer signe

```

35F8 E5      push hl
35F9 DDE1    pop  ix
35FB DD7E03 ld  a,(ix+03)      signe de la mantisse
35FE EE80    xor  80          inverser
3600 DD7703 ld  (ix+03),a
3603 C9      ret

```

***** FIX

```

3604 AF      xor  a
3605 DD9604 sub (ix+04)      exposant
3608 200A    jr   nz,3614    nombre non nul, alors à entier
360A 0604    ld  b,04
360C 77      ld  (hl),a      supprimer mantisse
360D 23      inc hl
360E 10FC    djnz 360C
3610 0E01    ld  c,01
3612 37      scf
3613 C9      ret

```

***** conversion virgule flottante à entier

```

3614 C6A0    add a,A0
3616 D0      ret  nc
3617 E5      push hl
3618 CD3D36  call 363D
361B AF      xor  a
361C B8      cp  b
361D 8F      adc a,a
361E B1      or  c
361F 4D      ld  c,l
3620 44      ld  b,h
3621 E1      pop  hl
3622 71      ld  (hl),c
3623 23      inc hl
3624 70      ld  (hl),b
3625 23      inc hl
3626 73      ld  (hl),e
3627 23      inc hl
3628 5F      ld  e,a
3629 7E      ld  a,(hl)
362A 72      ld  (hl),d
362B E680    and  80
362D 47      ld  b,a
362E 0E04    ld  c,04
3630 AF      xor  a
3631 B6      or  (hl)

```

ARITHMETIQUE A VIRGULE FLOTTANTE

3632	2005	jr	nz,3639
3634	2B	dec	hl
3635	0D	dec	c
3636	20F9	jr	nz,3631
3638	0C	inc	c
3639	7B	ld	a,e
363A	B7	or	a
363B	37	scf	
363C	C9	ret	
363D	FE21	cp	21
363F	3802	jr	c,3643
3641	3E21	ld	a,21
3643	5E	ld	e,(hl)
3644	23	inc	hl
3645	56	ld	d,(hl)
3646	23	inc	hl
3647	4E	ld	c,(hl)
3648	23	inc	hl
3649	66	ld	h,(hl)
364A	69	ld	l,c
364B	EB	ex	de,hl
364C	CBFA	set	7,d
364E	010000	ld	bc,0000
3651	180B	jr	365E
3653	4F	ld	c,a
3654	78	ld	a,b
3655	B5	or	l
3656	47	ld	b,a
3657	79	ld	a,c
3658	4D	ld	c,l
3659	6C	ld	l,h
365A	63	ld	h,e
365B	5A	ld	e,d
365C	1600	ld	d,00
365E	D608	sub	08
3660	30F1	jr	nc,3653
3662	C608	add	a,08
3664	C8	ret	z
3665	CB3A	srl	d
3667	CB1B	rr	e
3669	CB1C	rr	h
366B	CB1D	rr	l
366D	CB19	rr	c
366F	3D	dec	a
3670	20F3	jr	nz,3665
3672	C9	ret	
3673	14	inc	d
3674	15	dec	d
3675	F8	ret	m
3676	2017	jr	nz,368F
3678	57	ld	d,a
3679	7B	ld	a,e

ARITHMETIQUE A VIRGULE FLOTTANTE

```

367A B4      or   h
367B B5      or   l
367C B1      or   c
367D C8      ret   z
367E 7A      ld   a,d
367F D608    sub   08
3681 381C    jr    c,369F
3683 C8      ret   z
3684 53      ld   d,e
3685 5C      ld   e,h
3686 65      ld   h,l
3687 69      ld   l,c
3688 0E00    ld   c,00
368A 14      inc  d
368B 15      dec  d
368C 28F1    jr    z,367F
368E F8      ret   m
368F 3D      dec  a
3690 C8      ret   z
3691 CB21    sla  c
3693 CB15    rl   l
3695 CB14    rl   h
3697 CB13    rl   e
3699 CB12    rl   d
369B F28F36  jp   p,368F
369E C9      ret

369F AF      xor   a
36A0 C9      ret

```

***** conversion entier à virgule flottante

```

36A1 E5      push hl
36A2 DDE1    pop  ix
36A4 DD7004  ld   (ix+04),b      exposant
36A7 47      ld   b,a
36A8 5E      ld   e,(hl)
36A9 23      inc  hl
36AA 56      ld   d,(hl)
36AB 23      inc  hl
36AC 7E      ld   a,(hl)
36AD 23      inc  hl
36AE 66      ld   h,(hl)
36AF 6F      ld   l,a
36B0 EB      ex  de,hl
36B1 DD7E04  ld   a,(ix+04)      exposant
36B4 CD7336  call 3673
36B7 DD7704  ld   (ix+04),a      exposant
36BA CB21    sla  c
36BC 3013    jr   nc,36D1
36BE 2C      inc  l
36BF 2010    jr   nz,36D1
36C1 24      inc  h
36C2 200D    jr   nz,36D1
36C4 1C      inc  e
36C5 200A    jr   nz,36D1

```

ARITHMETIQUE A VIRGULE FLOTTANTE

```

36C7 14      inc    d
36C8 2007    jr     nz,36D1
36CA DD3404  inc    (ix+04)    exposant
36CD 281F    jr     z,36EE    dépassement
36CF 1680    ld     d,80
36D1 78      ld     a,b
36D2 F67F    or     7F
36D4 A2      and    d
36D5 DD7703  ld     (ix+03),a
36D8 DD7302  ld     (ix+02),e
36DB DD7401  ld     (ix+01),h
36DE DD7500  ld     (ix+00),l
36E1 DDE5    push  ix
36E3 E1      pop   hl
36E4 37     scf
36E5 C9     ret

```

*****dépassement par le bas, zéro

```

36E6 AF      xor    a
36E7 DD7704  ld     (ix+04),a    exposant
36EA 18F5    jr     36E1

```

*****dépassement, plus grand nombre positif
signe positif

```

36EC 0600    ld     b,00
36EE 78      ld     a,b
36EF F67F    or     7F
36F1 DD7703  ld     (ix+03),a    mantisse avec signe
36F4 F6FF    or     FF
36F6 DD7704  ld     (ix+04),a    exposant
36F9 DD7700  ld     (ix+00),a
36FC DD7701  ld     (ix+01),a
36FF DD7702  ld     (ix+02),a
3702 C9     ret
3703 C7     rst    0
3704 C7     rst    0
3705 C7     rst    0
3706 C7     rst    0
3707 C7     rst    0

```

ARITHMETIQUE AVEC ENTIERS

```

***** BDA3
3708 44          ld   b,h          ranger signe
3709 CDD137     call 37D1         former valeur absolue
370C 1802       jr    3710

***** BDA6
370E 0600       ld   b,00
3710 1E00       ld   e,00
3712 0E02       ld   c,02
3714 C9         ret

***** BDA9 accepter signe dans b
3715 7C         ld   a,h
3716 B7         or   a
3717 FA2037     jp   m,3720
371A B0         or   b          signe du résultat
371B FAD437     jp   m,37D4       inverser signe
371E 37         scf
371F C9         ret

3720 EE80       xor   80          inverser bit signe
3722 B5         or   l
3723 C0         ret   nz
3724 78         ld   a,b
3725 37         scf
3726 8F         adc   a,a
3727 C9         ret

***** BDAC Addition hl:=hl+de
3728 B7         or   a          annuler flag carry
3729 ED5A       adc   hl,de      addition
372B 37         scf
372C E0         ret   po        résultat positif?
372D F6FF       or   FF          fixer flags
372F C9         ret

***** BDB2 soustraction hl:=de-hl
3730 EB         ex   de,hl      échanger opérandes

***** BDAF soustraction hl:=hl-de
3731 B7         or   a          annuler flag carry
3732 ED52       sbc   hl,de      soustraction
3734 37         scf
3735 E0         ret   po        résultat positif?
3736 F6FF       or   FF          fixer flags
3738 C9         ret

***** BDB5 multiplication avec signe
3739 CD4537     call 3745       déterminer signe du résultat
373C CD5037     call 3750       multiplication sans signe
373F D21537     jp   nc,3715   accepter signe
3742 F6FF       or   FF
3744 C9         ret

```

ARITHMETIQUE AVEC ENTIERS

```

*****
3745 7C      ld  a,h      déterminer signe du résultat
3746 AA      xor  d        signe de hl
3747 47      ld  b,a      et signe de de
3748 EB      ex  de,hl    amener dans b
3749 CDD137  call 37D1     former valeur absolue de de
374C EB      ex  de,hl
374D C3D137  jp   37D1     former valeur absolue de hl

```

*****BDBE multiplication sans signe

```

3750 7C      ld  a,h
3751 B7      or   a
3752 2805    jr   z,3759
3754 7A      ld  a,d
3755 B7      or   a
3756 37      scf
3757 C0      ret  nz
3758 EB      ex  de,hl
3759 B5      or   l
375A C8      ret  z
375B 7A      ld  a,d
375C B3      or   e
375D 7D      ld  a,l
375E 6B      ld  l,e
375F 62      ld  h,d
3760 C8      ret  z
3761 FE03    cp   03
3763 3810    jr   c,3775
3765 37      scf
3766 8F      adc  a,a
3767 30FD    jr   nc,3766
3769 29      add  hl,hl
376A D8      ret  c
376B 87      add  a,a
376C 3002    jr   nc,3770
376E 19      add  hl,de
376F D8      ret  c
3770 FE80    cp   80
3772 20F5    jr   nz,3769
3774 C9      ret

3775 FE01    cp   01
3777 C8      ret  z
3778 29      add  hl,hl
3779 C9      ret

```

***** BDB8 division avec signe

```

377A CD8937  call 3789     division hl:= hl/de
377D DA1537  jp   c,3715  accepter signe
3780 C9      ret

```

***** BDBB MOD

```

3781 4C      ld  c,h      ranger signe
3782 CD8937  call 3789     division
3785 EB      ex  de,hl    reste dans hl

```

ARITHMETIQUE AVEC ENTIERS

3786	41	ld	b,c	rappeler signe
3787	18F4	jr	377D	et accepter
3789	CD4537	call	3745	déterminer signe du résultat
***** BDC1 Division hl:= h/de, de:= Rest				
378C	7A	ld	a,d	sans signe
378D	B3	or	e	diviseur zéro, alors terminer
378E	C8	ret	z	
378F	C5	push	bc	
3790	EB	ex	de,hl	
3791	0601	ld	b,01	
3793	7C	ld	a,h	
3794	B7	or	a	
3795	2009	jr	nz,37A0	
3797	7A	ld	a,d	
3798	BD	cp	l	
3799	3805	jr	c,37A0	
379B	65	ld	h,l	
379C	2E00	ld	l,00	
379E	0609	ld	b,09	
37A0	7B	ld	a,e	
37A1	95	sub	l	
37A2	7A	ld	a,d	
37A3	9C	sbc	a,h	
37A4	3805	jr	c,37AB	
37A6	04	inc	b	
37A7	29	add	hl,hl	
37A8	30F6	jr	nc,37A0	
37AA	3F	ccf		
37AB	3F	ccf		
37AC	78	ld	a,b	
37AD	44	ld	b,h	
37AE	4D	ld	c,l	
37AF	210000	ld	hl,0000	
37B2	3D	dec	a	
37B3	2003	jr	nz,37B8	
37B5	1817	jr	37CE	
37B7	29	add	hl,hl	
37B8	F5	push	af	
37B9	78	ld	a,b	
37BA	1F	rra		
37BB	47	ld	b,a	
37BC	79	ld	a,c	
37BD	1F	rra		
37BE	4F	ld	c,a	
37BF	7B	ld	a,e	
37C0	91	sub	c	
37C1	7A	ld	a,d	
37C2	98	sbc	a,b	
37C3	3805	jr	c,37CA	
37C5	57	ld	d,a	
37C6	7B	ld	a,e	
37C7	91	sub	c	

ARITHMETIQUE AVEC ENTIERS

```

37C8 5F      ld  e,a
37C9 2C      inc l
37CA F1      pop af
37CB 3D      dec a
37CC 20E9   jr  nz,37B7
37CE 37      scf
37CF C1      pop bc
37D0 C9      ret

```

```

***** former valeur absolue
37D1 7C      ld  a,h      tester signe
37D2 B7      or  a
37D3 F0      ret  p      positif, alors déjà terminé

```

```

***** BDC7 changement de signe hl
37D4 AF      xor  a
37D5 95      sub l
37D6 6F      ld  l,a
37D7 9C      sbc a,h
37D8 95      sub l
37D9 BC      cp  h
37DA 67      ld  h,a
37DB 37      scf
37DC C0      ret  nz
37DD FE01   cp  01
37DF C9      ret

```

```

***** BDCA SGN signe de hl
37E0 7C      ld  a,h
37E1 87      add a,a
37E2 9F      sbc a,a
37E3 D8      ret  c
37E4 B5      or  l
37E5 C8      ret  z
37E6 AF      xor  a
37E7 3C      inc a
37E8 C9      ret

```

```

***** BDC4 comparer hl <> de
37E9 7C      ld  a,h      signe de hl
37EA AA      xor  d      et signe de de
37EB 7C      lei a,h
37EC F2F437 jr  p,37F4   comparer nombres avec même signe
37EF 87      add a,a
37F0 9F      sbc a,a
37F1 D8      ret  c
37F2 3C      inc a
37F3 C9      ret

```

```

37F4 BA      cp  d
37F5 20F9   jr  nz,37F0
37F7 7D      ld  a,l
37F8 93      sub e
37F9 20F5   jr  nz,37F0
37FB C9      ret

```


BASIC 1.0

```

*****
C000 80      db      80      ROM-Header
C001 01      db      01      première Rom de devant
C002 00      db      00      Mark 1
C003 00      db      00      Version 0
C004 4CC0    dw      C04C    Modification 0
                               Adresse du nom

*****
C006 3100C0  ld      sp,C000  Initialisation du Basic
C009 CDCBBC  call     BCCB      pile à partir de C000
C00C CDC4F4  call     F4C4      KL ROM WALK
C00F DA0000  jp      c,0000    configurer la mémoire
                               trop peu de mémoire, alors Reset

C012 2100AC  ld      hl,AC00
C015 3600    ld      (hl),00
C017 061B    ld      b,1B
C019 23      inc     hl
C01A 36C9    ld      (hl),C9      'ret' de AC01 à AC1B
C01C 10FB    djnz   C019
C01E 213FC0  ld      hl,C03F      pointeur sur ' BASIC 1.0'
C021 CD37C3  call     C337      sortir
C024 AF      xor     a
C025 3200AC  ld      (AC00),a     suppr. flag pour 'ignorer espaces'
C028 CDCBDD  call     DDCB      adresse de ligne actuelle sur zéro
C02B CD84CA  call     CA84      supprimer numéro d'erreur
C02E CD97BD  call     BD97      RND Init
C031 CDD3C0  call     COD3      supprimer mode AUTO
C034 CD3EC1  call     C13E      NEW (instruction)
C037 11F000  ld      de,00F0     240
C03A CD06F7  call     F706      SYMBOL AFTER 240
C03D 1825    jr      C064      au mode READY

*****
C03F 20 42 41 53 49 43 20 31
C047 2E 30 0A 0A 00      'BASIC 1.0' LF,LF

*****
C04C 42 41 53 49 C3 00      'BAS1', 'C'+80H, 00H

*****
Instruction Basic5ED1T

```

BASIC 1.0

C052	CDE1CE	call	CEE1	aller chercher numéro de ligne
C055	CO	ret	nz	dans de
C056	3100CO	ld	sp,C000	initialiser la pile
C059	CD9AE7	call	E79A	chercher ligne Basic de (existe?)
C05C	CD63E1	call	E163	lister ligne Basic dans buffer
C05F	CD43CA	call	CA43	aller chercher ligne d'entrée
C062	3854	jr	c,COB8	

mode READY

C064	CD01AC	call	AC01	ret
C067	3100CO	ld	sp,C000	
C06A	CD62C1	call	C162	
C06D	CDD6DD	call	DDD6	aller chercher adresse de ligne
C070	DCB6BC	call	c,BCB6	SOUND HOLD
C073	CD48BB	call	BB48	KM DISARM BREAK
C076	CD86C3	call	C386	initialiser écran
C079	3A45AE	ld	a,(AE45)	programme protégé ?
C07C	B7	or	a	
C07D	C43EC1	call	nz,C13E	oui supprimer programme et variables
C080	3AAAAD	ld	a,(ADAA)	numéro ERROR
C083	D602	sub	O2	'Syntax error' ?
C085	2009	jr	nz,C090	non
C087	32AAAD	ld	(ADAA),a	numéro ERROR sur zéro
C08A	CDDFCA	call	CADF	aller chercher numéro de ligne
C08D	EB	ex	de,h1	de ligne ERROR
C08E	38C6	jr	c,C056	à l'instruction EDIT
C090	21CCCO	ld	hl,COCC	'Ready'
C093	CD41C3	call	C341	sortir
C096	CDCBDD	call	DDCB	adresse de ligne actuelle sur zéro
C099	3A1CAC	ld	a,(AC1C)	flag AUTO mis ?
C09C	B7	or	a	
C09D	2811	jr	z,COB0	non
C09F	CD02C1	call	C102	présenter prochain numéro de ligne
COA2	30CO	jr	nc,CO64	au mode READY
COA4	7E	ld	a,(h1)	
COA5	B7	or	a	
COA6	28F1	jr	z,C099	
COA8	CDD2E6	call	E6D2	convertir instruction en code
COAB	CD7AC1	call	C17A	interpréteur
COAE	18E9	jr	C099	

BASIC 1.0

```

*****
COB0 CD3BCA call CA3B aller chercher ligne d'entrée
COB3 30FB jr nc,COB0 'ESC' enfoncée, alors répéter
COB5 CD4EC3 call C34E sortir LF
COB8 CDBCE6 call E6BC convertir ligne en code interpréteur
COBB 3005 jr nc,COC2 instruction directe ?
COBD C47AC1 call nz,C17A
COCO 18D4 jr C096

COC2 CDBBDE call DEBB copier ligne dans buffer à partir de &40
COC5 CD53C4 call C453 autoriser interruption par 'Break'
COC8 2B dec hl
COC9 C374DD jp DD74 à la boucle de l'interpréteur

*****
COCC 52 65 61 64 79 0A 00 'Ready', LF, 00H

*****
COD3 AF xor a 0
COD4 1805 jr CODB

*****
COD6 221DAC ld (AC1D),hl numéro de ligne
COD9 3EFF ld a,FF
COdB 321CAC ld (AC1C),a fixer flag pour AUTO
CODE C9 ret

*****
CODF 110A00 ld de,000A Instruction Basic AUTO
COE2 2802 jr z,COE6 10, Defaut
COE4 FE2C cp 2C ','
COE6 C4E1CE call nz,CEE1 chercher No de ligne dans de
COE9 D5 push de
COEA 110A00 ld de,000A 10, Defaut
COED CD55DD call DD55 virgule suit?
COF0 DCE1CE call c,CEE1 oui, chercher No de ligne dans de
COF3 CD4ADD call DD4A fin de ligne, sinon 'Syntax error'
COF6 EB ex de,hl
COF7 221FAC ld (AC1F),hl ranger incrément AUTO
COFA E1 pop hl

```

BASIC 1.0

COFB	CDD6C0	call	C0D6	fixer flag pour mode AUTO
COFE	C1	pop	bc	
COFF	C396C0	jp	C096	
C102	2A1DAC	ld	hl,(AC1D)	No de ligne
C105	E5	push	hl	
C106	CD79EE	call	EE79	sortir No de ligne
C109	D1	pop	de	
C10A	CDA3E7	call	E7A3	chercher ligne
C10D	3E2A	ld	a,2A	'*'
C10F	3802	jr	c,C113	ligne existe ?
C111	3E20	ld	a,20	'6'
C113	CD56C3	call	C356	sortir
C116	CDD3C0	call	C0D3	supprimer mode AUTO
C119	CD3BCA	call	CA3B	aller chercher ligne d'entrée
C11C	D0	ret	nc	ESC enfoncée ?
C11D	CD4EC3	call	C34E	sortir LF
C120	E5	push	hl	
C121	2A1FAC	ld	hl,(AC1F)	No de ligne
C124	19	add	hl,de	plus incrément
C125	D4D6C0	call	nc,C0D6	fixer mode AUTO
C128	E1	pop	hl	
C129	37	scf		
C12A	C9	ret		

Instruction Basic NEW

C12B	C0	ret	nz	
C12C	CD3EC1	call	C13E	supprimer programme et variables
C12F	C364C0	jp	C064	au Mode READY

Instruction Basic CLEAR

C132	E5	push	hl	
C133	CD8CC1	call	C18C	
C136	CD5BC1	call	C15B	
C139	CD7AC1	call	C17A	
C13C	E1	pop	hl	
C13D	C9	ret		

supprimer programme et variables
début de la Ram libre

C13E	2A7FAE	ld	hl,(AE7F)	
------	--------	----	-----------	--

BASIC 1.0

C141	EB	ex	de,h1	
C142	2A7BAE	ld	h1,(AE7B)	HIMEM
C145	CDDAFF	call	FFDA	bc := h1 - de
C148	62	ld	h,d	
C149	6B	ld	l,e	
C14A	13	inc	de	
C14B	AF	xor	a	vider l'accu
C14C	77	ld	(h1),a	
C14D	EDB0	ldir		vider début Ram libre jusqu'à HIMEM
C14F	3245AE	ld	(AE45),a	supprimer flag pour progr. protégé
C152	CD76E6	call	E676	fin du programme := début du progr.
C155	CD8CC1	call	C18C	supprimer les variables
C158	CD6BC1	call	C16B	
C15B	CDADD2	call	D2AD	interrompre I/O cassette
C15E	AF	xor	a	
C15F	CD73BD	call	BD73	fixer mode RAD
C162	CDB3FB	call	FBB3	initialiser pile du descripteur
C165	CDFFD9	call	D9FD	
C168	C39DC1	jp	C19D	
C16B	CDE6DD	call	DDE6	TROFF
C16E	CDD3C0	call	C0D3	supprimer mode AUTO
C171	CDF2F1	call	F1F2	fixer TAB-Stops sur 13
C174	CD76E6	call	E676	fin de programme := début de programme
C177	CDB1D5	call	D5B1	restaurer pointeur de variable
C17A	CDD9CB	call	CBD9	supprimer ON-ERROR
C17D	CDABCB	call	CBAB	interdire CONT
C180	CDEDC8	call	C8ED	reset SOUND et Event
C183	CD8EF5	call	F58E	initialiser pile Basic
C186	CDD2D5	call	D5D2	supprimer flag pour FN
C189	C3E5DC	jp	DCE5	RESTORE
*****				supprimer variables
C18C	C5	push	bc	
C18D	È5	push	h1	
C18E	CDCAF5	call	F5CA	restaurer pointeur de chaîne
C191	CDAED5	call	D5AE	restaurer pointeur de variable
C194	CDFCDD	call	D5FC	Variables A-Z sur 'Real'
C197	CD89E9	call	E989	

BASIC 1.0

```
C19A E1      pop    hl
C19B C1      pop    bc
C19C C9      ret
```

```
C19D AF      xor    a
C19E CDAFC1  call   C1AF
C1A1 AF      xor    a
C1A2 E5      push   hl
C1A3 F5      push   af
C1A4 FE08    cp     08          < 8 ?
C1A6 DCB4BB call   c,BBB4     TXT STR SELECT
C1A9 F1      pop    af
C1AA 2121AC  ld     hl,AC21    numéro stream act.
C1AD 1804    jr     C1B3

C1AF E5      push   hl
C1B0 2122AC  ld     hl,AC22    canal d'entrée
C1B3 D5      push   de
C1B4 5F      ld     e,a
C1B5 7E      ld     a,(hl)
C1B6 73      ld     (hl),e
C1B7 D1      pop    de
C1B8 E1      pop    hl
C1B9 C9      ret

C1BA 3A21AC  ld     a,(AC21)   numéro stream act.
C1BD FE08    cp     08          imprimante?
C1BF C9      ret

C1C0 3A22AC  ld     a,(AC22)   canal d'entrée
C1C3 FE09    cp     09          cassette ?
C1C5 C9      ret

C1C6 CDE3C1  call   C1E3
C1C9 18D7    jr     C1A2

C1CB CDE3C1  call   C1E3
C1CE 18DF    jr     C1AF
```


BASIC 1.0

```

***** aller chercher numéro stream
C1D0 CDE3C1 call C1E3
C1D3 FE08 cp 08
C1D5 302E jr nc,C205 'Improper argument'
C1D7 CDA2C1 call C1A2
C1DA C1 pop bc
C1DB F5 push af
C1DC CDF9FF call FFF9 jp (bc) exécuter fonction
C1DF F1 pop af
C1E0 C3A2C1 jp C1A2

***** tester si numéro stream
C1E3 7E ld a,(hl)
C1E4 FE23 cp 23
C1E6 3E00 ld a,00 0 si default
C1E8 C0 ret nz
C1E9 CDF5C1 call C1F5 aller chercher numéro stream
C1EC F5 push af
C1ED CD55DD call DD55 virgule suit ?
C1F0 D44ADD call nc,DD4A non, alors fin de l'instruction?
C1F3 091 nop af

***** aller chercher numéro stream
C1F5 CD37DD call DD37 Tester si encore un caractère
C1F8 23 db 23 '#'
C1F9 3E0A ld a,0A 10, valeur maximale
C1FB C5 push bc
C1FC D5 push de
C1FD 47 ld b,a
C1FE CD67CE call CE67 aller chercher valeur 8 bits
C201 B8 cp b comparer avec b
C202 D1 pop de
C203 C1 pop bc
C204 D8 ret c inférieur à b, ok
C205 1E05 ld e,05 'Improper argument'
C207 C394CA jp CA94 sortir message d'erreur

***** Instruction Basic PAPER
C20A CDD0C1 call C1D0 aller chercher numéro stream

```

BASIC 1.0

C20D	0196BB	ld	bc,BB96	TXT SET PAPER
C210	1806	jr	C218	
*****				Instruction Basic PEN
C212	CDDOC1	call	C1D0	aller chercher numéro stream
C215	0190BB	ld	bc,BB90	TXT SET PEN
C218	CD4BC2	call	C24B	aller chercher argument < 16
C21B	E5	push	hl	
C21C	CDF9FF	call	FFF9	jp (bc) exécuter fonction
C21F	E1	pop	hl	
C220	C9	ret		
*****				Instruction Basic BORDER
C221	CD3CC2	call	C23C	aller chercher argument(s) < 32
C224	E5	push	hl	
C225	CD38BC	call	BC38	SCR SET BORDER
C228	E1	pop	hl	
C229	C9	ret		
*****				Instruction Basic INK
C22A	CD4BC2	call	C24B	aller chercher argument < 16
C22D	F5	push	af	
C22E	CD37DD	call	DD37	Tester si encore un caractère
C231	2C	db	2C	','
C232	CD3CC2	call	C23C	aller chercher argument(s) < 32
C235	F1	pop	af	
C236	E5	push	hl	
C237	CD32BC	call	BC32	SCR SET INK
C23A	E1	pop	hl	
C23B	C9	ret		
*****				aller chercher argument(s) < 32
C23C	CD44C2	call	C244	aller chercher argument < 32
C23F	41	ld	b,c	
C240	CD55DD	call	DD55	virgule suit?
C243	D0	ret	nc	non
C244	3E20	ld	a,20	32
C246	CDFBC1	call	C1FB	aller chercher argument < 32
C249	4F	ld	c,a	
C24A	C9	ret		

BASIC 1.0

```

*****
C24B 3E10    ld    a,10    aller chercher argument < 16
C24D 18AC    jr    C1FB    16
                    aller chercher argument < 16

*****
C24F 3E03    ld    a,03    Instruction Basic  MODE
C251 CDFBC1  call  C1FB    3
                    aller chercher argument < 3
C254 E5        push h1
C255 CDOEBC  call  BC0E    SCR SET MODE
C258 E1        pop  h1
C259 C9        ret

*****
C25A CDD0C1  call  C1D0    Instruction Basic  CLS
C25D 3E0C    ld    a,0C    aller chercher numéro stream
C25F C36EC3  jp    C36E    FF
                    sortir

*****
C262 0167C2  ld    bc,C267 VPOS
C265 1812    jr    C279

C267 3A21AC  ld    a,AC21  numéro stream act.
C26A FE08    cp    08      > 8 ?
C26C 3097    jr    nc,C205 'Improper argument'
C26E CD78BB  call  BB78    TXT GET CURSOR
C271 CD87BB  call  BB87    TXT VALIDATE
C274 7D        ld    a,1
C275 C9        ret

*****
C276 0190C2  ld    bc,C290 POS
C279 CDF5C1  call  C1F5    aller chercher valeur < 10
C27C CDA2C1  call  C1A2    Select Stream
C27F F5        push af
C280 CD37DD  call  DD37    Tester si encore un caractère
C283 29        db    29      ')'
C284 E5        push h1
C285 CDF9FF  call  FFF9    jp (bc) exécuter fonction
C288 CDOAFF  call  FFOA    accepter contenu accu comme nombre
                    entier

```

BASIC 1.0

C28B	E1	pop	hl	
C28C	F1	pop	af	
C28D	C3A2C1	jp	C1A2	Select Stream
*****				aller chercher position PRINT act.
C290	3A21AC	ld	a,(AC21)	Numéro stream act.
C293	FE08	cp	08	
C295	CADFC3	jp	z,C3DF	aller chercher position imprimante
C298	3A25AC	ld	a,(AC25)	aller chercher position cassette
C29B	D0	ret	nc	
C29C	C39CC3	jp	C39C	aller chercher position écran

C29F	3A21AC	ld	a,(AC21)	numéro stream act.
C2A2	FE08	cp	08	imprimante?
C2A4	280D	jr	z.C2B3	oui
C2A6	D0	ret	nc	cassette ?
C2A7	D5	push	de	
C2A8	E5	push	hl	
C2A9	CD69BB	call	BB69	TXT GET WINDOW
C2AC	7A	ld	a,d	
C2AD	94	sub	h	
C2AE	3C	inc	a	
C2AF	E1	pop	hl	
C2B0	D1	pop	de	
C2B1	37	scf		
C2B2	C9	ret		
C2B3	3A24AC	ld	a,(AC24)	WIDTH
C2B6	FEFF	cp	FF	
C2B8	C9	ret		
C2B9	E5	push	hl	
C2BA	CDBFC2	call	C2BF	
C2BD	E1	pop	hl	
C2BE	C9	ret		
C2BF	67	ld	h,a	
C2C0	CD9FC2	call	C29F	
C2C3	3F	ccf		

BASIC 1.0

```

C2C4 D8      ret      c
C2C5 6F      ld       l,a
C2C6 CD90C2  call     C290
C2C9 3D      dec      a
C2CA 37      scf
C2CB C8      ret      z
C2CC 84      add      a,h
C2CD 3F      ccf
C2CE D0      ret      nc
C2CF 3D      dec      a
C2D0 BD      cp       l
C2D1 C9      ret

```

Instruction Basic LOCATE
 aller chercher numéro stream
 aller chercher 2 valeurs 8 bits non nulles

```

C2D2 CDD0C1  call     C1D0
C2D5 CD27C3  call     C327

C2D8 E5      push     hl
C2D9 EB      ex      de,hl
C2DA 24      inc     h
C2DB 2C      inc     l
C2DC CD75BB  call     BB75
C2DF E1      pop     hl
C2E0 C9      ret

```

TXT SET CURSOR

Instruction Basic WINDOW

```

C2E1 7E      ld       a,(hl)
C2E2 FEE7    cp      E7
C2E4 2817    jr      z,C2FD
C2E6 CDD0C1  call     C1D0
C2E9 CD27C3  call     C327

C2EC D5      push     de
C2ED CD37DD  call     DD37
C2F0 2C      db      2C
C2F1 CD27C3  call     C327

C2F4 E3      ex      (sp),hl
C2F5 7A      ld      a,d
C2F6 55      ld      d,l

```

'SWAP'
 Aller chercher numéro stream
 Aller chercher 2 valeurs 8 bits non nulles
 Tester si encore un caractère
 ','
 aller chercher 2 valeurs 8 bits non nulles

BASIC 1.0

C2F7	6F	ld	l,a	
C2F8	CD66BB	call	BB66	TXT WIN ENABLE
C2FB	E1	pop	hl	
C2FC	C9	ret		

*****				WINDOW SWAP
C2FD	CD3FDD	call	DD3F	ignorer espaces
C300	CD12C3	call	C312	aller chercher argument < 8
C303	48	ld	c,b	
C304	CD55DD	call	DD55	virgule suit ?
C307	0600	ld	b,00	valeur défaut 0
C309	DC12C3	call	c,C312	oui, aller chercher argument < 8
C30C	E5	push	hl	
C30D	CDB7BB	call	BBB7	TXT SWAP STREAMS
C310	E1	pop	hl	
C311	C9	ret		

*****				aller chercher argument < 8
C312	3E08	ld	a,08	8
C314	CDFBC1	call	C1FB	aller chercher argument < 8
C317	47	ld	b,a	
C318	C9	ret		

*****				Instruction Basic TAG
C319	CDDOC1	call	C1D0	Aller chercher numéro stream
C31C	3EFF	ld	a,FF	
C31E	1804	jr	C324	

*****				Instruction Basic TAGOFF
C320	CDDOC1	call	C1D0	Aller chercher numéro stream
C323	AF	xor	a	
C324	C363BB	jp	BB63	TXT SET GRAPHIC

*****				aller chercher 2 valeurs 8 bits non nulles
C327	CD2FC3	call	C32F	aller chercher première valeur
C32A	53	ld	d,e	
C32B	CD37DD	call	DD37	Tester si encore un caractère
C32E	2C	db	2C	','
C32F	D5	push	de	

BASIC 1.0

```

C330 CD6DCE    call    CE6D        aller chercher valeur 8 bits non
                                nulle
C333 D1        pop     de
C334 5F        ld      e,a
C335 1D        dec     e
C336 C9        ret

*****
C337 3E84     ld      a,84        sortir chaîne
                                132
C339 3224AC    ld      (AC24),a    WIDTH sur 132
C33C E5         push   hl           adresse de début de la chaîne
C33D CD9DC1    call   C19D        sélectionner canal de sortie
C340 E1         pop    hl
C341 F5         push   af
C342 E5         push   hl
C343 7E         ld      a,(hl)      aller chercher un caractère
C344 23         inc    hl           augmenter pointeur
C345 B7         or     a           octet nul, donc fin de la chaîne
C346 C456C3    call   nz,C356     sortir un caractère
C349 20F8     jr     nz,C343     non nul, alors caractère suivant
C34B E1         pop    hl
C34C F1         pop    af
C34D C9        ret

*****
C34E F5         push   af          sortir LF
C34F 3E0A     ld      a,0A      LF
C351 CD56C3    call   C356     sortir
C354 F1         pop    af
C355 C9        ret

*****
C356 F5         push   af          sortir un caractère
C357 CD5CC3    call   C35C     sortir un caractère
C35A F1         pop    af
C35B C9        ret

C35C FE0A     cp     0A          LF ?
C35E 200E     jr     nz,C36E
C360 3A21AC    ld      a,(AC21)  numéro stream act.

```

BASIC 1.0

C363	FE08	cp	08	imprimante?
C365	CAA8C3	jp	z,C3A8	oui
C368	D2EAC3	jp	nc,C3EA	cassette
C36B	C392C3	jp	.C392	écran

***** sortir un caractère

C36E	F5	push	af	
C36F	C5	push	bc	
C370	4F	ld	c,a	caractère dans c
C371	CD77C3	call	C377	sortir
C374	C1	pop	bc	
C375	F1	pop	af	
C376	C9	ret		

***** sélectionner courant de sorti
numéro stream act.

C377	3A21AC	ld	a,(AC21)	
C37A	FE08	cp	08	
C37C	CAB5C3	jp	z,C3B5	imprimante ?
C37F	D2F8C3	jp	nc,C3F8	cassette ?
C382	79	ld	a,c	caractère dans accu
C383	C399C3	jp	C399	sortir caractère sur l'écran

***** initialiser écran

C386	AF	xor	a	
C387	CD63BB	call	BB63	TXT SET GRAPHIC
C38A	CD54BB	call	BB54	TXT VDU ENABLE
C38D	CD9CC3	call	C39C	Curseur dans position autorisée
C390	3D	dec	a	
C391	C8	ret	z	
C392	3E0D	ld	a,0D	CR
C394	CD99C3	call	C399	sortir
C397	3E0A	ld	a,0A	LF
C399	C35ABB	jp	BB5A	TXT OUTPUT

***** Curseur dans position autorisée

C39C	C5	push	bc	
C39D	E5	push	hl	
C39E	CD78BB	call	BB78	TXT GET CURSOR
C3A1	CD87BB	call	BB87	TXT VALIDATE
C3A4	7C	ld	a,h	

BASIC 1.0

```
C3A5 E1      pop    hl
C3A6 C1      pop    bc
C3A7 C9      ret
```

***** sortir CR & LF sur l'imprimante

```
C3A8 C5      push   bc
C3A9 0E0D    ld     c,0D      CR
C3AB CDB5C3  call  C3B5      sortir
C3AE 0E0A    ld     c,0A      LF
C3B0 CDB5C3  call  C3B5      sortir
C3B3 C1      pop    bc
C3B4 C9      ret
```

***** sortir caractère sur l'imprimante

```
C3B5 E5      push   hl
C3B6 79      ld     a,c
C3B7 EE0D    xor    0D      CR
C3B9 2813    jr     z,C3CE
C3BB 79      ld     a,c
C3BC FE20    cp     20      '6'
C3BE 3814    jr     c,C3D4  ne pas compter caract. de contrôle
C3C0 2A23AC  ld     hl,(AC23) position imprimante act. et WIDTH
C3C3 24      inc    h
C3C4 7D      ld     a,l
C3C5 2807    jr     z,C3CE
C3C7 BC      cp     h
C3C8 CCA8C3  call  z,C3A8
C3CB 3A23AC  ld     a,(AC23) position imprimante act.
C3CE 3C      inc    a
C3CF 2803    jr     z,C3D4
C3D1 3223AC  ld     (AC23),a position imprimante act.
```

BASIC 1.0

C3D4	E1	pop	h1	
C3D5	79	ld	a,c	
C3D6	CD2BBD	call	BD2B	MC PRINT CHAR
C3D9	D8	ret	c	sortie ok ?
C3DA	CD3CC4	call	C43C	interruption par 'ESC' ?
C3DD	18F6	jr	C3D5	

C3DF	3A23AC	ld	a,(AC23)	Position imprimante act.
C3E2	C9	ret		

C3E3	CD6DCE	call	CE6D	Instruction Basic WIDTH aller chercher valeur 8 bits non nulle
C3E6	3224AC	ld	(AC24),a	fixer WIDTH
C3E9	C9	ret		

C3EA	3E01	ld	a,01	nouvelle ligne sur cassette
C3EC	3225AC	ld	(AC25),a	position cassette sur 1
C3EF	3E0D	ld	a,0D	CR
C3F1	CD0DC4	call	C40D	sortir sur cassette
C3F4	3E0A	ld	a,0A	LF
C3F6	1815	jr	C40D	sortir sur cassette

C3F8	E5	push	h1	
C3F9	2125AC	ld	h1,AC25	position cassette
C3FC	79	ld	a,c	
C3FD	0601	ld	b,01	pour nouvelle ligne, position sur 1
C3FF	FE0D	cp	0D	CR
C401	2808	jr	z,C40B	
C403	FE20	cp	20	'6'
C405	3805	jr	c,C40C	ne pas compter caractères contrôle
C407	46	ld	b,(h1)	charger le compteur de caractères et l'augmenter
C408	04	inc	b	
C409	2801	jr	z,C40C	
C40B	70	ld	(h1),b	ranger nouvelle valeur compteur
C40C	E1	pop	h1	

BASIC 1.0

C40D	CD95BC	call	BC95	CAS OUT CHAR
C410	D8	ret	c	pas appuyé touche ESC ?
C411	C36BCB	jp	CB6B	'Break', mode READY

C414	C386BC	jp	BC86	CAS RETURN

				Variable réservée EOF
C417	E5	push	hl	
C418	CD89BC	call	BC89	CAS TEST EOF
C41B	28F4	jr	z,C411	ESC enfoncée ?
C41D	3F	ccf		
C41E	9F	sbc	a,a	
C41F	CD05FF	call	FF05	accepter signe comme nombre entier
C422	E1	pop	hl	
C423	C9	ret		

				aller chercher un caractère dans canal d'entrée
C424	3A22AC	ld	a,(AC22)	canal d'entrée
C427	FE09	cp	09	cassette ?
C429	CA80BC	jp	z,BC80	oui, CAS IN CHAR
C42C	CD09BB	call	BB09	KM READ CHAR
C42F	D8	ret	c	Touche enfoncée ?
C430	CD81BB	call	BB81	TXT CUR ON
C433	CD06BB	call	BB06	KM WAIT CHAR
C436	C384BB	jp	BB84	TXT CUR OFF

C439	C309BB	jp	BB09	KM READ CHAR

				Tester si interruption avec 'ESC'
C43C	CD09BB	call	BB09	KM READ CHAR
C43F	D0	ret	nc	
C440	FEFC	cp	FC	'Break' ?
C442	C0	ret	nz	
C443	C5	push	bc	
C444	D5	push	de	
C445	E5	push	hl	
C446	CD6FC4	call	C46F	attendre seconde frappe de touche
C449	DA6BCB	jp	c,CB6B	'ESC', alors interruption

BASIC 1.0

```

C44C CD53C4 call C453 autoriser interruption par 'Break'
C44F E1 pop hl
C450 D1 pop de
C451 C1 pop bc
C452 C9 ret

```

```

***** autoriser interruption par 'Break'
C453 E5 push hl
C454 115EC4 ld de,C45E Adresse de la routine Break-Event
C457 0EFD ld c,FD BASIC-ROM sélectionnée
C459 CD45BB call BB45 KM ARM BREAK
C45C E1 pop hl
C45D C9 ret

```

```

***** routine Break-Event
C45E E5 push hl
C45F CD09BB call BB09 KM READ CHAR
C462 3004 jr nc,C468 aucune touche enfoncée ?
C464 FEEF cp EF Break par 'ESC' ?
C466 20F7 jr nz,C45F ignorer touches frappées avant 'ESC'
C468 CD6FC4 call C46F attendre un second 'ESC'
C46B E1 pop hl
C46C C347C8 jp C847 Tester si ON BREAK GOSUB

```

```

***** attendre frappe d'une touche après 'ESC'
C46F CDB6BC call BCB6 SOUND HOLD
C472 F5 push af
C473 CD30C4 call C430 attendre frappe d'une touche
C476 FEEF cp EF Break par 'ESC' ?
C478 28F9 jr z,C473
C47A FEFC cp FC 'Break' ?
C47C 280B jr z,C489
C47E FE20 cp 20 '6' ?
C480 C40CBB call nz,BB0C non, ranger caractère KM CHAR RETURN
C483 F1 pop af
C484 DCB9BC call c,BCB9 SOUND CONTINUE
C487 B7 or a
C488 C9 ret

C489 F1 pop af

```

BASIC 1.0

C48A 37 scf
 C48B C9 ret

```

*****
C48C CD1AC5 call C51A      aller chercher 2 arguments
C48F C5      push bc
C490 D5      push de
C491 CD55DD call DD55      Virgule suit ?
C494 3018   jr      nc,C4AE Non
C496 CD1AC5 call C51A      aller chercher 2 arguments
C499 C5      push bc
C49A D5      push de
C49B CD37DD call DD37      Test auf nachfolgendes Zeichen
"amstrad 5"
C49E 2C      db      2C      ', '
C49F CD1AC5 call C51A      aller chercher 2 arguments
C4A2 C5      push bc
C4A3 E3      ex      (sp),hl
C4A4 CDD2BB call BBD2      GRA WIN HEIGHT
C4A7 E1      pop     hl
C4A8 D1      pop     de
C4A9 E3      ex      (sp),hl
C4AA CDCFBB call BBCF      GRA WIN WIDTH
C4AD E1      pop     hl
C4AE D1      pop     de
C4AF E3      ex      (sp),hl
C4B0 CDC9BB call BBC9      GRA SET ORIGIN
C4B3 E1      pop     hl
C4B4 C9      ret

C4B5 CD51DD call DD51      fin de l'instruction ?
C4B8 3806   jr      c,C4C0 oui
C4BA CD4BC2 call C24B      aller chercher argument < 16
C4BD CDE4BB call BBE4      GRA SET PAPER
C4C0 E5      push    hl
C4C1 CDDBBB call BBDB      GRA CLEAR WINDOW
C4C4 E1      pop     hl
C4C5 C9      ret
  
```

***** Instruction Basic DRAW

BASIC 1.0

C4C6	01F6BB	ld	bc, BBF6	GRA LINE ABSOLUTE
C4C9	180D	jr	C4D8	
*****				Instruction Basic DRAW
C4CF	01F9BB	ld	bc, BBF9	GRA LINE RELATIVE
C4CE	1808	jr	C4D8	
*****				Instruction Basic PLOT
C4D0	01EABB	ld	bc, BBEA	GRA PLOT ABSOLUTE
C4D3	1803	jr	C4D8	
*****				Instruction Basic PLOTR
C4D5	01EDBB	ld	bc, BBED	GRA PLOT RELATIVE
C4D8	C5	push	bc	
C4D9	CD1AC5	call	C51A	aller chercher 2 arguments
C4DC	CD55DD	call	DD55	virgule suit ?
C4DF	3006	jr	nc, C4E7	non
C4E1	CD4BC2	call	C24B	aller chercher argument < 16
C4E4	CDDEBB	call	BBDE	GRA SET PEN
C4E7	1828	jr	C511	
*****				TEST
C4E9	01F0BB	ld	bc, BBF0	GRA TEST ABSOLUTE
C4EC	1803	jr	C4F1	
*****				TESTR
C4EE	01F3BB	ld	bc, BBF3	GRA TEST RELATIVE
C4F1	C5	push	bc	
C4F2	CD1AC5	call	C51A	aller chercher 2 arguments
C4F5	CD37DD	call	DD37	tester si encore un caractère
C4F8	29	db	29	')'
C4F9	E3	ex	(sp), h1	
C4FA	C5	push	bc	
C4FB	E3	ex	(sp), h1	
C4FC	C1	pop	bc	
C4FD	CDF9FF	call	FFF9	jp (bc), exécuter fonction
C500	CDOAFF	call	FF0A	accepter contenu accu comme nombre entier
C503	E1	pop	h1	
C504	C9	ret		

BASIC 1.0

```

*****
C505 01COBB ld bc,BBC0 Instruction Basic MOVE
C508 1803 jr C50D GRA MOVE ABSOLUTE

*****
C50A 01C3BB ld bc,BBC3 Instruction Basic MOVER
C50D C5 push bc GRA MOVE RELATIVE
C50E CD1AC5 call C51A aller chercher 2 arguments
C511 E3 ex (sp),hl
C512 C5 push bc
C513 E3 ex (sp),hl
C514 C1 pop bc
C515 CDF9FF call FFF9 jp (bc), exécuter fonction
C518 E1 pop hl
C519 C9 ret

***** aller chercher deux arguments entiers dans de, bc
C51A CD86CE call CE86 aller chercher valeur 16-bits -32768
- +32767

C51D D5 push de
C51E CD37DD call DD37 Tester si encore un caractère
C521 2C db 2C ','
C522 CD86CE call CE86 aller chercher valeur 16-bits -32768
- +32767

C525 42 ld b,d
C526 4B ld c,e 2ème argument dans bc
C527 D1 pop de 1r argument
C528 C9 ret

*****
C529 CDB3D6 call D6B3 Instruction Basic FOR
C52C E5 push hl lire variable
C52D C5 push bc
C52E D5 push de
C52F CDC5C9 call C9C5 chercher NEXT correspondant
C532 222CAC ld (AC2C),hl ranger adresse
C535 D5 push de
C536 E5 push hl
C537 EB ex de,hl
C538 CD32C6 call C632 chercher boucle FOR-NEXT ouverte

```

BASIC 1.0

C53B	CCACF5	call	z,F5AC	trouvé, fixer pointeur de pile Basic
C53E	E1	pop	hl	
C53F	CD51DD	call	DD51	fin de l'instruction ?
C542	110000	ld	de,0000	zéro par défaut
C545	D486D6	call	nc,D686	non, aller chercher variable
C548	44	ld	b,h	
C549	4D	ld	c,l	
C54A	E1	pop	hl	
C54B	E3	ex	(sp),hl	
C54C	7A	ld	a,d	
C54D	B3	or	e	
C54E	C4B8FF	call	nz,FFB8	comparer hl <> de
C551	C2F6C5	jp	nz,C5F6	'Unexpected NEXT'
C554	EB	ex	de,hl	
C555	CDD2DD	call	DDD2	adresse de ligne actuelle dans hl
C558	E3	ex	(sp),hl	
C559	CDCEDD	call	DDCE	fixer adresse de ligne actuelle
C55C	E1	pop	hl	
C55D	F1	pop	af	
C55E	E3	ex	(sp),hl	
C55F	D5	push	de	
C560	C5	push	bc	
C561	E5	push	hl	
C562	010516	ld	bc,1605	22 octets, type 5 'Real'
C565	B9	cp	c	
C566	280B	jr	z,C573	
C568	010210	ld	bc,1002	16 octets, type 2 'Integer'
C56B	B9	cp	c	
C56C	2805	jr	z,C573	
C56E	1E0D	ld	e,0D	'Type mismatch'
C570	C394CA	jp	CA94	sortir message d'erreur
C573	78	ld	a,b	
C574	CDB0F5	call	F5B0	réserver place dans pile Basic
C577	73	ld	(hl),e	
C578	23	inc	hl	adresse de variable sur pile Basic
C579	72	ld	(hl),d	
C57A	23	inc	hl	
C57B	E3	ex	(sp),hl	
C57C	CD37DD	call	DD37	Tester si encore un caractère

BASIC 1.0

C57F	EF	db	EF	'='
C580	CDFBCE	call	CEFB	aller chercher expression
C583	79	ld	a,c	
C584	CDD7FE	call	FED7	comparer type de variable
C587	E5	push	hl	
C588	2127AC	ld	hl,AC27	mémoire provisoire pour variable FOR
C58B	CD62FF	call	FF62	copier variable dans hl
C58E	E1	pop	hl	
C58F	CD37DD	call	DD37	Tester si encore un caractère
C592	EC	db	EC	'T0'
C593	CDFBCE	call	CEFB	aller chercher expression
C596	E3	ex	(sp),hl	
C597	79	ld	a,c	
C598	CDD7FE	call	FED7	comparer type de variable
C59B	CD62FF	call	FF62	valeur finale sur pile Basic
C59E	EB	ex	de,hl	
C59F	E3	ex	(sp),hl	
C5A0	EB	ex	de,hl	
C5A1	210100	ld	hl,0001	un comme valeur STEP par défaut
C5A4	CD0DFF	call	FF0D	accepter nombre entier hl
C5A7	EB	ex	de,hl	
C5A8	7E	ld	a,(hl)	
C5A9	FEE6	cp	E6	'STEP'
C5AB	2006	jr	nZ,C5B3	
C5AD	CD3FDD	call	DD3F	ignorer espaces
C5B0	CDFBCE	call	CEFB	aller chercher expression
C5B3	79	ld	a,c	
C5B4	CDD7FE	call	FED7	Comparer type de variable
C5B7	E3	ex	(sp),hl	
"amstrad 6"				
C5B8	CD62FF	call	FF62	copier variable dans (hl)
C5BB	CDA3FD	call	FDA3	aller chercher signe
C5BE	EB	ex	de,hl	
C5BF	77	ld	(hl),a	signe de STEP sur pile Basic
C5C0	23	inc	hl	
C5C1	EB	ex	de,hl	
C5C2	E1	pop	hl	
C5C3	CD4ADD	call	DD4A	fin de l'instruction, sinon 'Syntax error'
C5C6	EB	ex	de,hl	

BASIC 1.0

C5C7	73	ld	(hl),e	
C5C8	23	inc	hl	Adresse de l'instruction FOR sur la pile Basic
C5C9	72	ld	(hl),d	
C5CA	23	inc	hl	
C5CB	EB	ex	de,hl	
C5CC	CDD2DD	call	DDD2	Adresse de ligne act. dans hl
C5CF	EB	ex	de,hl	
C5D0	73	ld	(hl),e	
C5D1	23	inc	hl	Adresse de ligne de FOR sur pile Basic
C5D2	72	ld	(hl),d	
C5D3	23	inc	hl	
C5D4	D1	pop	de	
C5D5	73	ld	(hl),e	
C5D6	23	inc	hl	Adresse de l'instruction NEXT sur pile Basic
C5D7	72	ld	(hl),d	
C5D8	23	inc	hl	
C5D9	ED5B2CAC	ld	de,(AC2C)	
C5DD	73	ld	(hl),e	
C5DE	23	inc	hl	adresse de l'instruction NEXT sur pile Basic
C5DF	72	ld	(hl),d	
C5E0	23	inc	hl	
C5E1	70	ld	(hl),b	&10 ou &16 pour Integer/Real sur pile
C5E2	D1	pop	de	
C5E3	2127AC	ld	hl,AC27	pointeur sur mémoire provisoire
C5E6	CD66FF	call	FF66	aller chercher variable FOR
C5E9	AF	xor	a	
C5EA	3226AC	ld	(AC26),a	Flag pour premier parcours
C5ED	E1	pop	hl	
C5EE	CDCEDD	call	DDCE	fixer adresse de ligne act.
C5F1	2A2CAC	ld	hl,(AC2C)	
C5F4	180A	Jr	C600	à l'instruction NEXT
C5F6	1E01	ld	e,01	'Unexpected NEXT'
C5F8	C394CA	jp	CA94	sortir message d'erreur

BASIC 1.0

```

***** Instruction Basic NEXT
C5FB 3EFF ld a,FF
C5FD 3226AC ld (AC26),a flag pour additionner incrément
C600 EB ex de,h1
C601 CD32C6 call C632 chercher boucle FOR-NEXT ouverte
C604 20F0 jr nz,C5F6 'Unexpected NEXT'
C606 EB ex de,h1
C607 CDACF5 call F5AC fixer pointeur de pile Basic
C60A EB ex de,h1
C60B E5 push hl
C60C CD61C6 call C661 Tester si fin de boucle
C60F 280F jr z,C620
C611 F1 pop af
C612 23 inc hl
C613 5E ld e,(hl)
C614 23 inc hl pointeur de programme dans de
C615 56 ld d,(hl)
C616 23 inc hl
C617 7E ld a,(hl)
C618 23 inc hl adresse de ligne dans hl
C619 66 ld h,(hl)
C61A 6F ld l,a
C61B CDCEDD call DDCE fixer adresse de ligne act.
C61E EB ex de,h1
C61F C9 ret

C620 010500 ld bc,0005 pointeur de pile Basic
C623 09 add hl,bc plus 5
C624 5E ld e,(hl)
C625 23 inc hl pointeur de programme dans 'NEXT'
C626 56 ld d,(hl)
C627 E1 pop hl
C628 CDACF5 call F5AC fixer pointeur de pile Basic
C62B EB ex de,h1
C62C CD55DD call DD55 virgule suit ?
C62F 38CF jr c,C600 oui, prochaine boucle NEXT
C631 C9 ret

***** chercher boucle FOR-NEXT ouverte
C632 2A8B80 ld hl,(B08B) Pointeur de pile Basic

```

BASIC 1.0

C635	E5	push	hl	
C636	2B	dec	hl	
C637	46	ld	b, (hl)	
C638	23	inc	hl	
C639	7D	ld	a, l	
C63A	90	sub	b	
C63B	6F	ld	l, a	
C63C	9F	sbc	a, a	
C63D	84	add	a, h	
C63E	67	ld	h, a	
C63F	E3	ex	(sp), hl	
C640	78	ld	a, b	
C641	FE07	cp	07	'WHILE-WEND' ?
C643	2819	jr	z, C65E	
C645	FE10	cp	10	Integer 'FOR-NEXT' ?
C647	2804	jr	z, C64D	
C649	FE16	cp	16	Real 'FOR-NEXT' ?
C64B	200D	jr	nz, C65A	
C64D	E5	push	hl	
C64E	2B	dec	hl	
C64F	2B	dec	hl	
C650	7E	ld	a, (hl)	
C651	2B	dec	hl	
C652	6E	ld	l, (hl)	
C653	67	ld	h, a	
C654	CDB8FF	call	FFB8	comparer hl <> de
C657	E1	pop	hl	
C658	2004	jr	nz, C65E	
C65A	EB	ex	de, hl	
C65B	E1	pop	hl	
C65C	78	ld	a, b	
C65D	C9	ret		
C65E	E1	pop	hl	
C65F	18D4	jr	C635	
C661	5E	ld	e, (hl)	
C662	23	inc	hl	
C663	56	ld	d, (hl)	
C664	23	inc	hl	

BASIC 1.0

C665	FE10	cp	10	Integer ?
C667	282D	jr	z,C696	
C669	E5	push	hl	
C66A	010500	ld	bc,0005	Type sur 'Real'
C66D	79	ld	a,c	
C66E	EB	ex	de,hl	
C66F	CD4BFF	call	FF4B	accepter variable et type
C672	E1	pop	hl	
C673	3A26AC	ld	a,(AC26)	flag pour premier parcours
C676	B7	or	a	
C677	2810	jr	z,C689	oui, sauter addition
C679	E5	push	hl	
C67A	09	add	hl,bc	
C67B	CDCCFC	call	FCCC	additionner valeur STEP
C67E	E1	pop	hl	
C67F	E5	push	hl	
C680	2B	dec	hl	
C681	56	ld	d,(hl)	
C682	2B	dec	hl	
C683	5E	ld	e,(hl)	
C684	EB	ex	de,hl	
C685	CD62FF	call	FF62	copier variable dans (hl)
C688	E1	pop	hl	
C689	E5	push	hl	
C68A	0E05	ld	c,05	
C68C	CD09FD	call	FD09	comparaison arithmétique
C68F	E1	pop	hl	
C690	010A00	ld	bc,000A	10
C693	09	add	hl,bc	
C694	96	sub	(hl)	
C695	C9	ret		
C696	E5	push	hl	
C697	EB	ex	de,hl	
C698	5E	ld	e,(hl)	

BASIC 1.0

```

C699 23      inc    h1
C69A 56      ld     d,(h1)
C69B 3A26AC  ld     a,(AC26)    premier parcours ?
C69E B7       or     a
C69F 2816    jr     z,C6B7      oui, sauter addition
C6A1 E3      ex     (sp),h1
C6A2 E5      push   h1
C6A3 23      inc    h1
C6A4 23      inc    h1
C6A5 7E      ld     a,(h1)
C6A6 23      inc    h1      aller chercher valeur STEP dans h1
C6A7 66      ld     h,(h1)
C6A8 6F      ld     l,a
C6A9 CDACBD  call   BDAC      Integer-Addition h1 := h1 + de
C6AC 1E06    ld     e,06      'Overflow'
C6AE D294CA  jp     nc,CA94    sortir message d'erreur

C6B1 EB      ex     de,h1
C6B2 E1      pop    h1
C6B3 E3      ex     (sp),h1
C6B4 72      ld     (h1),d
C6B5 2B      dec    h1
C6B6 73      ld     (h1),e
C6B7 E1      pop    h1
C6B8 7E      ld     a,(h1)
C6B9 23      inc    h1
C6BA E5      push   h1
C6BB 66      ld     h,(h1)
C6BC 6F      ld     l,a
C6BD EB      ex     de,h1
C6BE CDC4BD  call   BDC4      comparer Integer
C6C1 E1      pop    h1
C6C2 23      inc    h1
C6C3 23      inc    h1
C6C4 23      inc    h1
C6C5 96      sub    (h1)
C6C6 C9      ret

***** Instruction Basic IF
C6C7 CDFBCE  call   CEFB      aller chercher expression

```

BASIC 1.0

C6CA	FEA0	cp	A0	'GOTO'
C6CC	2804	Jr	z,C6D2	
C6CE	CD37DD	call	DD37	Tester si encore un caractère
C6D1	EB	db	EB	'THEN'
C6D2	E5	push	hl	
C6D3	CDA3FD	call	FDA3	aller chercher signe
C6D6	E1	pop	hl	
C6D7	CC9FE8	call	z,E89F	chercher fin de ligne ou branchement ELSE
C6DA	C8	ret	z	
C6DB	CD51DD	call	DD51	fin de l'instruction ?
C6DE	D8	ret	c	oui
C6DF	FE1E	cp	1E	Numéro de ligne ?
C6E1	2805	Jr	z,C6E8	Oui, zum GOTO-Befehl
C6E3	FE1D	cp	1D	adresse de ligne ?
C6E5	C2ABDD	Jp	nz,DDAB	non, exécuter instruction Basic

C6E8	CD67E7	call	E767	Instruction Basic GOTO aller chercher adresse de ligne
C6EB	EB	ex	de,hl	accepter adresse comme pointeur de programme
C6EC	C9	ret		

C6ED	CD67E7	call	E767	Instruction Basic GOSUB aller chercher adresse de ligne
C6F0	CDEFE8	call	E8EF	instruction DATA, ignorer le reste de la ligne d'instruction
C6F3	EB	ex	de,hl	
C6F4	0E00	ld	c,00	marque pour 'GOSUB' normal
C6F6	E5	push	hl	ranger adresse du sous-programme
C6F7	3E06	ld	a,06	6 octets
C6F9	CDB0F5	call	F5B0	réserver place dans pile Basic
C6FC	71	ld	(hl),c	zéro
C6FD	23	inc	hl	
C6FE	73	ld	(hl),e	
C6FF	23	inc	hl	Adresse instruction après 'GOSUB'
C700	72	ld	(hl),d	sur pile Basic
C701	23	inc	hl	
C702	EB	ex	de,hl	
C703	CDD2DD	call	DDD2	actuelle adresse de ligne dans hl

BASIC 1.0

```

C706 EB      ex      de,h1
C707 73      ld      (h1),e
C708 23      inc     h1      Adresse de ligne sur pile Basic
C709 72      ld      (h1),d
C70A 23      inc     h1
C70B 3606    ld      (h1),06    marque pour 'GOSUB'
C70D E1      pop     hl      pointeur de programme sur sous-
                             programme
C70E C9      ret

```

Instruction Basic RETURN

```

C70F C0      ret     nz
C710 CD2EC7  call    C72E    chercher GOSUB sur Pile Basic
C713 CDACF5  call    F5AC    réinitialiser pointeur de pile Basic
C716 4E      ld      c,(h1)   octet-marque
C717 23      inc     hl
C718 5E      ld      e,(h1)
C719 23      inc     hl      aller chercher adresse de
C71A 56      ld      d,(hl)   l'instruction après 'GOSUB' dans de
C71B 23      inc     hl
C71C 7E      ld      a,(hl)
C71D 23      inc     hl      Adresse de ligne dans hl
C71E 66      ld      h,(hl)
C71F 6F      ld      l,a
C720 CDCEDD  call    DDCE    fixer numéro de ligne actuel
C723 EB      ex      de,hl   pointeur de programme actuel dans hl
C724 79      ld      a,c      octet distinctif
C725 FE01    cp      01      inférieur à 1 ?
C727 D8      ret     c      oui, GOSUB normal
C728 CAA4C8  jp      z,C8A4   un, alors GOSUB après AFTER/EVERY
C72B C3B6C8  jp      C8B6

```

```

C72E 2A8BB0   ld      hl,(B08B)  Pointeur de pile Basic
C731 2B      dec     hl
C732 7E      ld      a,(hl)    aller chercher marque das pile
                             Basic
C733 F5      push   af
C734 7D      ld      a,l
C735 96      sub    (hl)

```


BASIC 1.0

```

C736 6F      ld      l,a
C737 9F      sbc     a,a
C738 84      add     a,h      restaurer pointeur de pile Basic
C739 67      ld      h,a
C73A 23      inc     hl
C73B F1      pop     af
C73C FE06   cp      06      'GOSUB'
C73E C8      ret     z
C73F B7      or      a
C740 20EF    jr      nz,C731
C742 1E03    ld      e,03      'Unexpected RETURN'
C744 C394CA  jp     CA94      Sortir message d'erreur

```

Instruction Basic WHILE

```

C747 E5      push    hl
C748 CD18CA  call   CA18      chercher WEND correspondant
C74B E5      push    hl      ranger adresse
C74C EB      ex     de,hl
C74D 222EAC  ld     (AC2E),hl  Adresse de ligne pour WHILE-WEND
C750 CDB8C7  call   C7B8
C753 CCACF5  call   z,F5AC      fixer pointeur de pile Basic
C756 3E07    ld     a,07      7 octets
C758 CDB0F5  call   F5B0      réserver place dans pile Basic
C75B EB      ex     de,hl
C75C CDD2DD  call   DDD2      actuelle adresse de ligne dans hl
C75F EB      ex     de,hl
C760 73      ld     (hl),e
C761 23      inc    hl      Adresse de ligne sur pile Basic
C762 72      ld     (hl),d
C763 23      inc    hl
C764 D1      pop    de
C765 73      ld     (hl),e
C766 23      inc    hl      Adresse après 'WEND' sur pile Basi
C767 72      ld     (hl),d
C768 23      inc    hl
C769 EB      ex     de,hl
C76A E3      ex     (sp),hl
C76B EB      ex     de,hl
C76C 73      ld     (hl),e
C76D 23      inc    hl      Adresse de condition 'WHILE'

```

BASIC 1.0

```

C76E 72      ld      (hl),d      sur pile Basic
C76F 23      inc     hl
C770 3607    ld      (hl),07      marque pour 'WHILE'
C772 EB      ex      de,hl
C773 D1      pop     de
C774 182A    jr      C7A0          tester condition 'WHILE'

```

***** Instruction Basic WEND

```

C776 C0      ret     nz
C777 EB      ex      de,hl
C778 CDB8C7  call    C7B8
C77B 1E1E    ld      e,1E          'Unexpected WEND'
C77D C294CA  jp      nz,CA94      Sortir message d'erreur

C780 E5      push   hl
C781 110700   ld      de,0007
C784 19      add    hl,de
C785 CDACF5  call    F5AC          hl comme pointeur de pile Basic
C788 CDD2DD   call    DDD2          actuelle adresse de ligne dans hl
C78B 222EAC  ld      (AC2E),hl    Adresse de ligne pour WHILE-WEND
C78E E1      pop     hl
C78F 5E      ld      e,(hl)
C790 23      inc    hl
C791 56      ld      d,(hl)
C792 23      inc    hl
C793 EB      ex      de,hl
C794 CDCEDD  call    DDCE          fixer actuelle adresse de ligne
C797 EB      ex      de,hl
C798 5E      ld      e,(hl)
C799 23      inc    hl
C79A 56      ld      d,(hl)
C79B 23      inc    hl
C79C 7E      ld      a,(hl)
C79D 23      inc    hl
C79E 66      ld      h,(hl)
C79F 6F      ld      l,a
C7A0 D5      push   de
C7A1 CDFBCE  call    CEFB          aller chercher expression
C7A4 E5      push   hl
C7A5 CDA3FD  call    FDA3          aller chercher signe

```

BASIC 1.0

```

C7A8 E1      pop    hl
C7A9 D1      pop    de
C7AA C0      ret    nz      condition remplie ?
C7AB 2A2EAC ld    hl,(AC2E) Adresse de ligne pour WHILE-WEND
C7AE CDCEDD  call   DDCE      fixer comme actuelle adresse de
                           ligne
C7B1 3E07    ld     a,07
C7B3 CDA0F5  call   F5A0      libérer place dans pile Basic
C7B6 EB      ex     de,hl
C7B7 C9      ret

```

```

C7B8 2A8B0   ld     hl,(B08B)  Pointeur de pile Basic
C7BB 2B      dec    hl
C7BC E5      push   hl
C7BD 7D      ld     a,l
C7BE 96      sub    (hl)
C7BF 6F      ld     l,a
C7C0 9F      sbc   a,a
C7C1 84      add   a,h
C7C2 67      ld     h,a
C7C3 23      inc   hl
C7C4 E3      ex    (sp),hl
C7C5 7E      ld     a,(hl)
C7C6 FE10   cp    10      Integer 'FOR-NEXT'
C7C8 2816   jr    z,C7E0
C7CA FE16   cp    16      Real 'FOR-NEXT'
C7CC 2812   jr    z,C7E0
C7CE FE07   cp    07      'WHILE-WEND'
C7D0 200C   jr    nz,C7DE
C7D2 2B      dec    hl
C7D3 2B      dec    hl
C7D4 2B      dec    hl
C7D5 7E      ld     a,(hl)
C7D6 2B      dec    hl
C7D7 6E      ld     l,(hl)
C7D8 67      ld     h,a
C7D9 CDB8FF  call   FFB8      comparer hl <> de
C7DC 2002   jr    nz,C7E0
C7DE E1      pop    hl

```

BASIC 1.0

C7DF C9 ret

C7E0 E1 pop hl
C7E1 18D8 jr C7BB

```
***** Instruction Basic ON
C7E3 FE9C     cp       9C       'ERROR'
C7E5 CAE5CB   jp       z,CBE5
C7E8 CD67CE   call     CE67       aller chercher valeur 8 bits
C7EB 4F       ld       c,a       comme compteur dans c
C7EC 46       ld       b,(hl)   aller chercher token
C7ED 78       ld       a,b
C7EE FEA0     cp       A0       'GOTO'
C7F0 2805     jr       z,C7F7
C7F2 CD37DD   call     DD37       Tester si encore un caractère
C7F5 9F       db       9F       'GOSUB'
C7F6 2B       dec       hl
C7F7 0D       dec       c       diminuer compteur
C7F8 78       ld       a,b       Token dans a
C7F9 CAABDD   jp       z,DDAB   exécuter instruction Basic
C7FC CD3FDD   call     DD3F       Ignorer les espaces
C7FF CDE1CE   call     CEE1       aller chercher numéro de ligne dans
de
C802 FE2C     cp       2C       ','
C804 28F1     jr       z,C7F7   prochain numéro de ligne
C806 C9       ret
```

```
***** Traitement des Events (AFTER/EVERY)
C807 AF       xor       a
C808 3230AC   ld       (AC30),a
C80B CDFBBC   call     BCFB       KL NEXT SYNC
C80E 301D     jr       nc,C82D   y a-t-il un évènement ?
C810 47       ld       b,a       ranger priorité
C811 3A30AC   ld       a,(AC30)
C814 E67F     and       7F       annuler bit 7
C816 3230AC   ld       (AC30),a
C819 C5       push    bc
C81A E5       push    hl       Adresse du bloc Event
C81B CDFEBC   call     BCFE       KL DO SYNC
C81E E1       pop       hl
```

BASIC 1.0

```

C81F C1      pop      bc
C820 3A30AC ld      a,(AC30)
C823 17      rla
C824 F5      push     af
C825 78      ld      a,b
C826 D401BD call   nc,BD01      KL DONE SYNC
C829 F1      pop      af
C82A 17      rla
C82B 30DE    jr      nc,C80B      prochain Event
C82D 3A30AC ld      a,(AC30)
C830 E604    and     04
C832 C453C4 call   nz,C453      autoriser interruption par 'ESC'
C835 2A34AE ld      hl,(AE34)   Adresse de l'instruction actuelle
C838 3A30AC ld      a,(AC30)
C83B E603    and     03
C83D C8      ret     z
C83E 1F      rra
C83F DA6BCB jp      c,CB6B      'Break'
C842 23      inc     hl
C843 F1      pop      af
C844 C393DD  jp      DD93      à la boucle de l'interpréteur

```

```

C847 2236AC ld      (AC36),hl
C84A 3E04    ld      a,04
C84C 3050    jr      nc,C89E
C84E 2A34AC ld      hl,(AC34)   adresse ON-BREAK
C851 7C      ld      a,h
C852 B5      or      l
C853 C4D6DD call   nz,DDD6      Numéro de ligne dans hl
C856 3E41    ld      a,41
C858 3044    jr      nc,C89E      mode direct ?
C85A 1131AC ld      de,AC31
C85D 0E02    ld      c,02
C85F 1825    jr      C886

```

```

C861 D5      push     de
C862 CD37DD call   DD37      Tester si encore un caractère
C865 9F      db      9F      'GOSUB'

```

BASIC 1.0

C866	CD67E7	call	E767	aller chercher adresse de ligne
C869	42	ld	b,d	
C86A	4B	ld	c,e	
C86B	CD61DD	call	DD61	ignorer espace, TAB et LF
C86E	D1	pop	de	
C86F	E5	push	hl	
C870	210A00	ld	hl,000A	10
C873	19	add	hl,de	
C874	71	ld	(hl),c	
C875	23	inc	hl	
C876	70	ld	(hl),b	
C877	E1	pop	hl	
C878	C9	ret		

Routine Event

C879	23	inc	hl	
C87A	23	inc	hl	
C87B	23	inc	hl	
C87C	EB	ex	de,hl	
C87D	CDD6DD	call	DDD6	aller chercher numéro de ligne mode direct ?
C880	3E40	ld	a,40	
C882	301A	jr	nc,C89E	oui
C884	0E01	ld	c,01	octet distinctif pour AFTER/EVERY
GOSUB				
C886	D5	push	de	
C887	CDF6C6	call	C6F6	instruction GOSUB
C88A	2A34AE	ld	hl,(AE34)	Adresse de l'instruction actuelle
C88D	EB	ex	de,hl	
C88E	E1	pop	hl	
C88F	70	ld	(hl),b	
C890	23	inc	hl	
C891	73	ld	(hl),e	
C892	23	inc	hl	
C893	72	ld	(hl),d	
C894	23	inc	hl	
C895	5E	ld	e,(hl)	
C896	23	inc	hl	
C897	56	ld	d,(hl)	
C898	EB	ex	de,hl	

BASIC 1.0

C899	2234AE	ld	(AE34),h1	Adresse de l'instruction actuelle
C89C	3EC2	ld	a,C2	
C89E	2130AC	ld	h1,AC30	
C8A1	B6	or	(h1)	
C8A2	77	ld	(h1),a	
C8A3	C9	ret		
C8A4	7E	ld	a,(h1)	
C8A5	23	inc	h1	
C8A6	5E	ld	e,(h1)	
C8A7	23	inc	h1	
C8A8	56	ld	d,(h1)	
C8A9	D5	push	de	
C8AA	01F7FF	ld	bc,FFF7	
C8AD	09	add	h1,bc	
C8AE	CD01BD	call	BD01	KL DONE SYNC
C8B1	E1	pop	h1	
C8B2	F1	pop	af	
C8B3	C374DD	jp	DD74	à la boucle de l'interpréteur
C8B6	7E	ld	a,(h1)	
C8B7	2A36AC	ld	h1,(AC36)	
C8BA	01FCFF	ld	bc,FFFC	
C8BD	09	add	h1,bc	
C8BE	CD01BD	call	BD01	KL DONE SYNC
C8C1	CD53C4	call	C453	autoriser interruption par 'Break'
C8C4	2A32AC	ld	h1,(AC32)	
C8C7	F1	pop	af	
C8C8	C374DD	jp	DD74	à la boucle de l'interpréteur
*****				Instruction Basic ON BREAK
C8CB	FECE	cp	CE	'STOP'
C8CD	110000	ld	de,0000	valeur défaut 0 pour STOP
C8D0	2808	Jr	z,C8DA	
C8D2	CD37DD	call	DD37	Tester si encore un caractère
C8D5	9F	db	9F	'GOSUB'
C8D6	CD67E7	call	E767	aller chercher adresse de ligne
C8D9	2B	dec	h1	
C8DA	ED5334AC	ld	(AC34),de	adresse ON-BREAK
C8DE	C33FDD	jp	DD3F	Ignorer les espaces

BASIC 1.0

```

*****
C8E1 E5      push  h1
C8E2 CD04BD  call  BD04      KL EVENT DISABLE
C8E5 E1      pop   h1
C8E6 C9      ret

*****
C8E7 E5      push  h1
C8E8 CD07BD  call  BD07      KL EVENT ENABLE
C8EB E1      pop   h1
C8EC C9      ret

*****
C8ED CDA7BC  call  BCA7      Reset SOUND et Event
C8F0 215CAC  ld    h1,AC5C  SOUND RESET
C8F3 0604    ld    b,04      adresse de base du bloc Event
C8F5 E5      push  h1        4 Timer
C8F6 CDECBC  call  BCEC      KL DEL TICKER
C8F9 E1      pop   h1
C8FA 111200  ld    de,0012  18
C8FD 19      add   h1,de     additionner
C8FE 10F5    djnz  C8F5     prochain Timer
C900 CD48BB  call  BB48     KM DISARM BREAK
C903 CDF5BC  call  BCF5     KL SYNC RESET
C906 210000  ld    h1,0000
C909 2234AC  ld    (AC34),h1  supprimer adresse ON-BREAK
C90C CD53C4  call  C453     autoriser interruption par 'Break'
C90F 2138AC  ld    h1,AC38
C912 110503  ld    de,0305
C915 010008  ld    bc,0800
C918 CD24C9  call  C924
C91B 2162AC  ld    h1,AC62  Adresse du bloc Event
C91E 110B04  ld    de,040B
C921 010102  ld    bc,0201
C924 C5      push  bc
C925 D5      push  de
C926 0EFD    ld    c,FD     sélect. BASIC-ROM
C928 1179C8  ld    de,C879  Adresse de routine Event
C92B CDEFBC  call  BCEF     KL INIT EVENT
C92E D1      pop   de

```


BASIC 1,0

```

C92F D5      push   de
C930 1600    ld      d,00
C932 19      add     hl,de
C933 D1      pop     de
C934 C1      pop     bc
C935 79      ld      a,c
C936 B7      or      a
C937 2803    jr      z,C93C
C939 78      ld      a,b
C93A 87      add     a,a
C93B 47      ld      b,a
C93C 15      dec     d
C93D 20E5    jr      nz,C924
C93F C9      ret

```

```

***** Instruction Basic ON SQ
C940 CD37DD  call   DD37      Tester si encore un caractère
C943 28      db      28      '('
C945 CD67CE  call   CE67      aller chercher valeur 8-bits
C947 F5      push   hl
C948 CD5DC9  call   C95D      calculer adresse de Sound-Queue
C94B B7      or      a        supérieur à 4 ?
C94C 201E    jr      nz,C96C  'Improper argument'
C94E CD37DD  call   DD37      Tester si encore un caractère
C951 29      db      29      ')'
C952 CD61C8  call   C861      aller chercher 'GOSUB' et adresse
C955 F1      pop     af
C956 E5      push   hl
C957 EB      ex     de,hl
C958 CDB0BC  call   BCBO      SOUND ARM EVENT
C95B E1      pop     hl
C95C C9      ret

```

```

***** calculer adresse de Sound-Queue
C95D 1F      rra
C95E 1138AC  ld      de,AC38  Bit 0 mis ?
C961 D8      ret     c
C962 1F      rra      Bit 1 mis ?
C963 1144AC  ld      de,AC44
C966 D8      ret     c

```

BASIC 1.0

C967	1F	rra		Bit 2 mis ?
C968	1150AC	ld	de,AC50	
C96B	D8	ret	c	
C96C	1E05	ld	e,05	'Improper argument'
C96E	C394CA	jp	CA94	Sortir message d'erreur
*****				Instruction Basic AFTER
C971	CD7CCE	call	CE7C	aller chercher valeur 16-bits 0 - 32767
C974	010000	ld	bc,0000	Recharge Count sur zéro
C977	1805	jr	C97E	
*****				Instruction Basic EVERY
C979	CD7CCE	call	CE7C	aller chercher valeur 16 bits 0 - 32767
C97C	42	ld	b,d	comme Count et
C97D	4B	ld	c,e	Recharge Count
C97E	D5	push	de	
C97F	C5	push	bc	
C980	CD55DD	call	DD55	suit virgule ?
C983	110000	ld	de,0000	valeur par défaut zéro
C986	DC86CE	call	c,CE86	oui, aller chercher valeur entière avec signe
C989	EB	ex	de,h1	
C98A	CDB1C9	call	C9B1	aller chercher dans timer# adresse du bloc Event
C98D	E5	push	h1	
C98E	010600	ld	bc,0006	additionner 6 octets pour bloc ticker
C991	09	add	h1,bc	

BASIC 1.0

```

C992 EB      ex      de,h1
C993 CD61C8 call    C861      aller chercher 'GOSUB' et adresse
C996 D1      dop     de
C997 C1      pop     bc
C998 E3      ex      (sp),h1
C999 EB      ex      de,h1
C99A CDE9BC call    BCE9      KL ADD TICKER
C99D E1      pop     h1
C99E C9      ret

```

```

***** fonction BASIC REMAIN
C99F CD8DFE call    FE8D      CINT
C9A2 CDB1C9 call    C9B1      aller chercher adresse du bloc EVENT
C9A5 CDECBC call    BCEC      KL DEL TICKER
C9A8 3803   jr      c,C9AD   trouvé ?
C9AA 110000 ld      de,0000   non, zéro
C9AD EB      ex      de,h1
C9AE C30DFF jp      FF0D      accepter nombre entier dans h1

```

```

***** calculer adresse du bloc EVENT
C9B1 7C      ld      a,h
C9B2 B7      or      a          Hi-Byte différent de zéro ?
C9B3 20B7   jr      nz,C96C   oui, 'Improper argument'
C9B5 7D      ld      a,l
C9B6 FE04   cp      04          supérieur égal à 4 ?
C9B8 30B2   jr      nc,C96C   oui, 'Improper argument'
C9BA 87      add     a,a
C9BB 87      add     a,a
C9BC 87      add     a,a          * 18
C9BD 85      add     a,l
C9BE 87      add     a,a
C9BF 6F      ld      l,a
C9C0 015CAC ld      bc,AC5C   adresse de base table EVENT
C9C3 09      add     hl,bc     plus Offset
C9C4 C9      ret

```

```

***** chercher NEXT correspondant
C9C5 EB      ex      de,h1
C9C6 CDD2DD call    DDD2      adresse de ligne act. dans h1
C9C9 EB      ex      de,h1

```

BASIC 1.0

C9CA	2B	dec	hl	
C9CB	0601	ld	b,01	compteur pour imbrication
C9CD	0E1A	ld	c,1A	numéro d'erreur pour 'NEXT missing'
C9CF	CD23E9	call	E923	
C9D2	E5	push	hl	
C9D3	CD3FDD	call	DD3F	ignorer les espaces
C9D6	FEBO	cp	B0	'NEXT'
C9D8	2808	jr	z,C9E2	
C9DA	E1	pop	hl	
C9DB	FE9E	cp	9E	'FOR'
C9DD	20EE	jr	nz,C9CD	
C9DF	04	inc	b	augmenter imbrication
C9E0	18EB	jr	C9CD	continuer à chercher
C9E2	F1	pop	af	
C9E3	EB	ex	de,hl	
C9E4	E5	push	hl	
C9E5	CDD2DD	call	DDD2	adresse de ligne actuelle dans hl
C9E8	E3	ex	(sp),hl	
C9E9	CDCEDD	call	DDCE	fixer adresse de ligne actuelle
C9EC	EB	ex	de,hl	
C9ED	05	dec	b	diminuer imbrication
C9EE	2824	jr	z,CA14	trouvé 'NEXT' correspondant ?
C9F0	CD3FDD	call	DD3F	ignorer les espaces
C9F3	280E	jr	z,CA03	fin de ligne ?
C9F5	C5	push	bc	
C9F6	D5	push	de	
C9F7	CD86D6	call	D686	chercher variable
C9FA	D1	pop	de	
C9FB	C1	pop	bc	
C9FC	CD55DD	call	DD55	virgule suit ?
C9FF	3002	jr	nc,CA03	non
CA01	10F2	djnz	C9F5	sinon prochaine variable après 'NEXT'
CA03	2B	dec	hl	
CA04	78	ld	a,b	trouvé 'NEXT' correspondant ?
CA05	B7	or	a	
CA06	280C	jr	z,CA14	oui
CA08	EB	ex	de,hl	
CA09	CDD2DD	call	DDD2	adresse de ligne actuelle dans hl

BASIC 1.0

CA0C	E3	ex	(sp),h1	
CA0D	CDCEDD	call	DDCE	fixer adresse de ligne actuelle
CA10	E1	pop	h1	
CA11	EB	ex	de,h1	
CA12	18B9	jr	C9CD	continuer à chercher
CA14	D1	pop	de	
CA15	C33FDD	jp	DD3F	ignorer les espaces
*****				chercher WEND correspondant
CA18	2B	dec	h1	
CA19	EB	ex	de,h1	
CA1A	CDD2DD	call	DDD2	adresse de ligne actuelle dans h1
CA1D	EB	ex	de,h1	
CA1E	0600	ld	b,00	compteur pour imbrication
CA20	04	inc	b	
CA21	0E1D	ld	c,1D	numéro d'erreur pour 'WEND missing'
CA23	CD23E9	call	E923	
CA26	E5	push	h1	
CA27	CD3FDD	call	DD3F	ignorer les espaces
CA2A	E1	pop	h1	
CA2B	FED6	cp	D6	'WHILE'
CA2D	28F1	jr	z,CA20	augmenter imbrication
CA2F	FED5	cp	D5	'WEND'
CA31	20EE	jr	nz,CA21	
CA33	10EC	djnz	CA21	diminuer imbrication
CA35	CD3FDD	call	DD3F	ignorer les espaces
CA38	C33FDD	jp	DD3F	ignorer les espaces
*****				aller chercher ligne d'entrée
CA3B	21A4AC	ld	h1,ACA4	pointeur sur buffer d'entrée
CA3E	3600	ld	(h1),00	vider contenu buffer
CA40	C33ABD	jp	BD3A	aller chercher ligne d'entrée
*****				éditer ligne
CA43	21A4AC	ld	h1,ACA4	pointeur sur buffer d'entrée
CA46	CD3ABD	call	BD3A	éditer ligne
CA49	C34EC3	jp	C34E	sortir LF
*****				aller chercher ligne d'entrée dans cassette

BASIC 1.0

CA4C	C5	push	bc	
CA4D	D5	push	de	
CA4E	21A4AC	ld	h1,ACA4	pointeur sur buffer d'entrée
CA51	E5	push	h1	
CA52	0601	ld	b,01	
CA54	0E00	ld	c,00	
CA56	CD80BC	call	BC80	CAS IN CHAR
CA59	CA6BCB	jp	z,CB6B	
CA5C	3022	Jr	nc,CA80	
CA5E	77	ld	(h1),a	
CA5F	FE0D	cp	OD	CR
CA61	2817	Jr	z,CA7A	
CA63	0E00	ld	c,00	
CA65	FE0A	cp	0A	LF
CA67	2006	Jr	nz,CA6F	
CA69	78	ld	a,b	,
CA6A	3D	dec	a	
CA6B	28E7	Jr	z,CA54	
CA6D	0EFF	ld	c,FF	
CA6F	78	ld	a,b	
CA70	B7	or	a	
CA71	1E17	ld	e,17	'Line too long'
CA73	CA94CA	jp	z,CA94	sortir message d'erreur
CA76	23	inc	h1	
CA77	04	inc	b	
CA78	18DC	Jr	CA56	
CA7A	79	ld	a,c	
CA7B	B7	or	a	
CA7C	20D8	Jr	nz,CA56	
CA7E	77	ld	(h1),a	
CA7F	37	scf		
CA80	E1	pop	h1	
CA81	D1	pop	de	
CA82	C1	pop	bc	
CA83	C9	ret		
*****				supprimer numéro d'erreur
CA84	AF	xor	a	

BASIC 1.0

*****				fixer numéro d'erreur
CA85	32AAAD	ld	(ADAA),a	numéro d'erreur
CA88	CDD2DD	call	DDD2	adresse de ligne actuelle dans hl
CA8B	22A6AD	ld	(ADA6),hl	ERROR-Line
CA8E	C9	ret		
*****				instruction Basic ERROR
CA8F	CD6DCE	call	CE6D	aller chercher valeur 8 bits non nulle
CA92	C0	ret	nz	
CA93	5F	ld	e,a	numéro d'erreur dans e
*****				sortir message d'erreur
CA94	C004AC	call	AC04	ret
CA97	7B	ld	a,e	
CA98	CD85CA	call	CA85	ranger numéro et ligne d'erreur
CA9B	2A34AE	ld	hl,(AE34)	Adresse de l'instruction actuelle
CA9E	22A8AD	ld	(ADA8),hl	pointeur de programme dans ERROR
CAA1	CDB0CB	call	CB00	ranger adresse de ligne et pointeur de programme
CAA4	3100C0	ld	sp,C000	pointeur de pile sur C000
CAA7	2A32AE	ld	hl,(AE32)	mémoire pour pointeur de pile Basic
CAA	CDACF5	call	F5AC	réinitialiser pointeur de pile Basic
CAAD	CDB3FB	call	FBB3	initialiser pile du descripteur
CAB0	CDFD9	call	D9FD	vider AE29 et AE2B
CAB3	CDDFCA	call	CADF	aller chercher numéro de ligne de la ligne d'erreur
CAB6	2AAFAD	ld	hl,(ADAF)	Adresse de la routine ON-ERROR
CAB9	EB	ex	de,hl	
CABA	21B1AD	ld	hl,ADB1	Flag pour traitement d'erreur
CABD	300C	jr	nc,CACB	
CABF	7A	ld	a,d	
CACO	B3	or	e	
CAC1	2808	jr	z,CACB	
CAC3	A6	and	(hl)	
CAC4	2005	jr	nz,CACB	
CAC6	35	dec	(hl)	
CAC7	EB	ex	de,hl	
CAC8	C393DD	jp	DD93	à la boucle de l'interpréteur

BASIC 1.0

CACB	3600	ld	(h1),00	
CACD	3AAAAD	ld	a,(ADAA)	numéro ERROR
CADO	CD45CC	call	CC45	fixer pointeur sur message d'erreur
CAD3	2AA6AD	ld	h1,(ADA6)	Adresse de la ligne ERROR
CAD6	CDCEDD	call	DDCE	fixer adresse de ligne actuelle
CAD9	CD36CB	call	CB36	
CADC	C364C0	jp	C064	au mode READY

CADF	2AA6AD	ld	h1,(ADA6)	Adresse de la ligne ERROR
CAE2	CDD9DD	call	DDD9	aller chercher numéro de ligne dans h1
CAE5	D8	ret	c	
CAE6	210000	ld	h1,0000	
CAE9	C9	ret		

CAEA	D5	push	de	
CAEB	E5	push	h1	
CAEC	2113CD	ld	h1,CD13	pointeur sur
CAEF	1E0B	ld	e,0B	'Division by zero'
CAF1	1807	Jr	CAFA	
CAF3	D5	push	de	
CAF4	E5	push	h1	
CAF5	21B9CC	ld	h1,CCB9	pointeur sur
CAF8	1E06	ld	e,06	'Overflow'
CAFA	F5	push	af	
CAFB	E5	push	h1	
CAFC	2AAFAD	ld	h1,(ADAF)	Adresse de la routine ON-ERROR
CAFF	7C	ld	a,h	
CB00	B5	or	l	
CB01	E1	pop	h1	
CB02	C294CA	jp	nz,CA94	sortir message d'erreur
CB05	AF	xor	a	
CB06	CDA2C1	call	C1A2	
CB09	F5	push	af	
CB0A	CD41C3	call	C341	sortir chaîne
CB0D	CD4EC3	call	C34E	sortir LF

BASIC 1.0

```

CB10 F1      pop    af
CB11 CDA2C1  call   C1A2
CB14 F1      pop    af
CB15 E1      pop    hl
CB16 D1      pop    de
CB17 C9      ret

CB18 CD86C3  call   C386      initialiser écran
CB1B 2123CB  ld     hl,CB23    sortir 'Undefined line'
CB1E CD48CB  call   CB48      sortir numéro de ligne
CB21 181D    jr     CB40      sortir 'in numéro de ligne'

*****
CB23 55 6E 64 65 66 69 6E 65
CB2B 64 20 6C 69 6E 65 20 00      'Undefined line '

*****
CB33 114FCB  ld     de,CB4F    pointeur sur 'Break'
CB36 CD9DC1  call   C19D
CB39 CD86C3  call   C386      initialiser écran
CB3C EB      ex     de,hl
CB3D CD41C3  call   C341      sortir
CB40 CDD6DD  call   DDD6      aller chercher No de ligne dans hl
CB43 D0      ret    nc      mode direct ?
CB44 EB      ex     de,hl
CB45 2155CB  ld     hl,CB55    pointeur sur ' in '
CB48 CD41C3  call   C341      sortir
CB4B EB      ex     de,hl
CB4C C379EE  jp     EE79      sortir numéro de ligne

*****
CB4F 42 72 65 61 6B 00      'Break'
CB55 20 69 6E 20 00      ' in '

*****
instruction Basic STOP
CB5A C0      ret    nz
CB5B E5      push  hl
CB5C CD33CB  call   CB33      'Break in numéro de ligne'
CB5F E1      pop    hl
CB60 CD93CB  call   CB93

```

BASIC 1.0

```

CB63 182B jr CB90 au mode READY

***** instruction Basic END
CB65 C0 ret nz
CB66 CD93CB call CB93
CB69 181C jr CB87

CB6B CD33CB call CB33
CB6E 2A34AE ld hl,(AE34) Adresse de l'instruction actuelle
CB71 CDB0CB call CB80
CB74 181A Jr CB90

CB76 CDD6DD call DDD6 mode direct ?
CB79 3012 jr nc,CB8D oui
CB7B CDABCB call CBAB
CB7E 3AB1AD ld a,(ADB1) encore en traitement d'erreur ?
CB81 B7 or a
CB82 1E13 ld e,13 'RESUME missing'
CB84 C294CA Jp nz,CA94 oui, sortir message d'erreur

CB87 CD98D2 call D298
CB8A CDA1D2 call D2A1
CB8D CDCBDD call DDCB adresse de ligne actuelle sur zéro
CB90 C364C0 Jp C064 au mode READY

*****
CB93 EB ex de,hl
CB94 CDD6DD call DDD6 aller chercher No de ligne dans hl
CB97 EB ex de,hl
CB98 D0 ret nc mode direct ?
CB99 7E ld a,(hl)
CB9A FE01 cp 01
CB9C 280B jr z,CBA9
CB9E 23 inc hl
CB9F 7E ld a,(hl)
CBA0 23 inc hl
CBA1 B6 or (hl)
CBA2 2807 Jr z,CBAB
CBA4 23 inc hl
CBA5 CDCEDD call DDCE fixer adresse de ligne actuelle

```

BASIC 1.0

CBA8	23	inc	hl	
CBA9	1805	jr	CB80	
CBAB	210000	ld	hl,0000	
CBAE	180C	jr	CB8C	
CB80	EB	ex	de,hl	
CB81	CDD6DD	call	DDD6	mode direct ?
CB84	D0	ret	nc	oui
CB85	CDD2DD	call	DDD2	adresse de ligne dans hl
CB88	22ADAD	ld	(ADAD),hl	adresse de ligne après interruption
CB8B	EB	ex	de,hl	
CB8C	22ABAD	ld	(ADAB),hl	pointeur de programme après interruption
CB8F	C9	ret		
***** instruction Basic CONT				
CBC0	C0	ret	nz	
CBC1	2AABAD	ld	hl,(ADAB)	pointeur de programme après interruption
CBC4	7C	ld	a,h	
CBC5	B5	or	l	
CBC6	1E11	ld	e,11	'Cannot Continue'
CBC8	CA94CA	jp	z,CA94	sortir message d'erreur
CBCB	E5	push	hl	
CBCC	2AADAD	ld	hl,(ADAD)	adresse de ligne après interruption
CB8F	CDCEDD	call	DDCE	fixer adresse de ligne actuelle
CBD2	CDB9BC	call	BCB9	SOUND CONTINUE
CBD5	E1	pop	hl	
CBD6	C374DD	jp	DD74	à la boucle de l'interpréteur

CBD9	AF	xor	a	
CBDA	32B1AD	ld	(ADB1),a	supprimer flag pour en traitement d'erreur
CBDD	110000	ld	de,0000	
CBE0	ED53AFAD	ld	(ADAF),de	Adresse de la routine ON-ERROR
CBE4	C9	ret		

BASIC 1.0

```

*****
CBE5 CD3FDD call DD3F ignorer les espaces
CBE8 CD37DD call DD37 Tester si encore un caractère
CBEB A0 db A0 'GOTO'
CBEC CDE1CE call CEE1 aller chercher No de ligne dans de
CBEF E5 push hl
CBFO CD9AE7 call E79A chercher ligne Basic de
CBF3 22AFAD ld (ADAF),hl Adresse de la routine ON-ERROR
CBF6 E1 pop hl
CBF7 C9 ret

*****
CBF8 CDDDCB call CBDD instruction Basic ON ERROR GOTO 0
CBFB 3AB1AD ld a,(ADB1) adresse de ligne actuelle sur zéro
CBFE B7 or a en traitement d'erreur ?
CBFF C8 ret z non
CC00 C3A4CA jp CAA4

*****
CC03 2814 jr z,CC19 instruction Basic RESUME
CC05 FE80 cp B0 'NEXT'
CC07 2817 jr z,CC20
CC09 CD67E7 call E767 aller chercher adresse de ligne
CC0C CD4ADD call DD4A fin de l'instruction, sinon 'Syntax
error'

CC0F D5 push de
CC10 CD2BCC call CC2B supprimer flags ERROR
CC13 E1 pop hl
CC14 23 inc hl
CC15 F1 pop af
CC16 C393DD jp DD93 à la boucle de l'interpréteur

CC19 CD2BCC call CC2B supprimer flags ERROR
CC1C F1 pop af
CC1D C374DD jp DD74 à la boucle de l'interpréteur

CC20 CD3FDD call DD3F ignorer les espaces
CC23 C0 ret nz
CC24 CD2BCC call CC2B supprimer flags ERROR
CC27 23 inc hl

```

BASIC 1.0

CC28	C3EFE8	jp	E8EF	ignorer reste de la ligne
CC2B	3AB1AD	ld	a,(ADB1)	en traitement d'erreur ?
CC2E	B7	or	a	
CC2F	1E14	ld	e,14	'Unexpected RESUME'
CC31	CA94CA	jp	z,CA94	non, sortir message d'erreur
CC34	AF	xor	a	
CC35	32AAAD	ld	(ADAA),a	annuler numéro ERROR
CC38	32B1AD	ld	(ADB1),a	supprimer flag pour en traitement d'erreur
CC3B	2AA6AD	ld	hl,(ADA6)	Adresse de la ligne ERROR
CC3E	CDCEDD	call	DDCE	comme adresse de ligne actuelle
CC41	2AA8AD	ld	hl,(ADA8)	pointeur de programme après ERROR
CC44	C9	ret		
*****				fixer pointeur sur message d'erreur
CC45	115BCC	ld	de,CC5B	adresse de base des messages d'erreur
CC48	FE1F	cp	1F	31, numéro d'erreur maxi +1
CC4A	D0	ret	nc	>=, alors 0 'Unknown error'
CC4B	B7	or	a	
CC4C	C8	ret	z	0, alors terminé
CC4D	47	ld	b,a	numéro d'erreur dans b
CC4E	1A	ld	a,(de)	
CC4F	13	inc	de	lire caractère
CC50	B7	or	a	Jusqu'à 0, fin d'un message
CC51	20FB	jr	nz,CC4E	
CC53	05	dec	b	prochain message
CC54	20F8	jr	nz,CC4E	pas encore message voulu ?
CC56	1A	ld	a,(de)	
CC57	B7	or	a	
CC58	28EB	jr	z,CC45	
CC5A	C9	ret		
*****				messages d'erreur
CC5B	55 6E 6B 6E 6F 77 6E 20 65			
CC64	72 72 6F 72 00			0 Unknown error
CC69	55 6E 65 78 70 65 63 74 65			
CC72	64 20 4E 45 58 54 00			1 Unexpected NEXT

BASIC 1.0

CC79 53 79 6E 74 61 78 20 65 72	
CC82 72 6F 72 00	2 Syntax error
CC86 55 6E 65 78 70 65 63 74 65	
CC8F 64 20 52 45 54 55 52 4E 00	3 Unexpected RETURN
CC98 44 41 54 41 20 65 78 68 61	
CCA1 75 73 74 65 64 00	4 DATA exhausted
CCA7 49 6D 70 72 6F 70 65 72 20	
CCB0 61 72 67 75 6D 65 6E 74 00	5 Improper argument
CCB9 4F 76 65 72 66 6C 6F 77 00	6 Overflow
CCC2 4D 65 6D 6F 72 79 20 66 75	
CCCB 6C 6C 00	7 Memory full
CCCE 4C 69 6E 65 20 64 6F 65 73	
CCD7 20 6E 6F 74 20 65 78 69 73	
CCEO 74 00	8 Line does not exist
CCE2 53 75 62 73 63 72 69 70 74	'
CCEB 20 6F 75 74 20 6F 66 20 72	
CCF4 61 6E 67 65 00	9 Subscript out of range
CCF9 41 72 72 61 79 20 61 6C 72	
CD02 65 61 64 79 20 64 69 6D 65	
CD0B 6E 73 69 6F 6E 65 64 00	10 Array already dimensioned
CD13 44 69 76 69 73 69 6F 6E 20	
CD1C 62 79 20 7A 65 72 6F 00	11 Division by zero
CD24 49 6E 76 61 6C 69 64 20 64	
CD2D 69 72 65 63 74 20 63 6F 6D	
CD35 6D 61 6E 64 00	12 Invalid direct command
CD3B 54 79 70 65 20 6D 69 73 6D	
CD44 61 74 63 68 00	13 Type mismatch
CD49 53 74 72 69 6E 67 20 73 70	
CD52 61 63 65 20 66 75 6C 6C 00	14 String space full
CD5B 53 74 72 69 6E 67 20 74 6F	
CD64 6F 20 6C 6F 6E 67 00	15 String too long
CD6B 53 74 72 69 6E 67 20 65 78	
CD74 70 72 65 73 73 69 6F 6E 20	
CD7D 74 6F 6F 20 63 6F 6D 70 6C	
CD86 65 78 00	16 String expression too complex
CD89 43 61 6E 6E 6F 74 20 43 4F	
CD92 4E 54 69 6E 75 65 00	17 Cannot CONTINUE
CD99 55 6E 6B 6E 6F 77 6E 20 75	
CDA2 73 65 72 20 66 75 6E 63 74	
CDAB 69 6F 6E 00	18 Unknown user function

BASIC 1 0

CDAF 52 45 53 55 4D 45 20 6D 69	
CDB8 73 73 69 6E 67 00	19 RESUME missing
CDBE 55 6E 65 78 70 65 63 74 65	
CDC7 64 20 52 45 53 55 4D 45 00	20 Unexpected RESUME
CDD0 44 69 72 65 63 74 20 63 6F	
CDD9 6D 6D 61 6E 64 20 66 6F 75	
CDE2 6E 64 00	21 Direct command found
CDE5 4F 70 65 72 61 6E 64 20 6D	
CDEE 69 73 73 69 6E 67 00	22 Operand missing
CDF5 4C 69 6E 65 20 74 6F 6F 20	
CDFE 6C 6F 6E 67 00	23 Line too long
CE03 45 4F 46 20 6D 65 74 00	24 EOF met
CE0B 46 69 6C 65 20 74 79 70 65	
CE14 20 65 72 72 6F 72 00	25 File type error
CE1B 4E 45 58 54 20 6D 69 73 73	
CE24 69 6E 67 00	26 NEXT missing
CE28 46 69 6C 65 20 61 6C 72 65	
CE31 61 64 79 20 6F 70 65 6E 00	27 File already open
CE3A 55 6E 6B 6E 6F 77 6E 20 63	
CE43 6F 6D 6D 61 6E 64 00	28 Unknown command
CE4A 57 45 4E 44 20 6D 69 73 73	
CE53 69 6E 67 00	29 WEND missing
CE57 55 6E 65 78 70 65 63 74 65	
CE60 64 20 57 45 4E 44 00	30 Unexpected WEND

CE67 CD86CE call CE86	aller chercher valeur 8 bits aller chercher valeur entière avec signe
CE6A F5 push af	
CE6B 1808 jr CE75	

CE6D CD86CE call CE86	aller chercher valeur 8 bits différente de zéro aller chercher valeur entière avec signe
CE70 F5 push af	
CE71 7A ld a,d	
CE72 B3 or e	zéro ?
CE73 2836 jr z,CEAB	alors 'improper argument'
CE75 7A ld a,d	
CE76 B7 or a	Hi-Byte différent de zéro ?

BASIC 1.0

```
CE77 2032 Jr nz,CEAB alors 'Improper argument'
CE79 F1 pop af
CE7A 7B ld a,e
CE7B C9 ret
```

```
***** aller chercher valeur 16 bits 0 à 32767
CE7C CD86CE call CE86 aller chercher valeur entière avec
signe
CE7F F5 push af
CE80 7A ld a,d
CE81 17 rla Bit 15 mis ?
CE82 3827 Jr c,CEAB alors 'Improper argument'
CE84 F1 pop af
CE85 C9 ret
```

```
***** aller chercher valeur entière avec signe
CE86 CDFBCE call CEFB aller chercher expression
CE89 F5 push af
CE8A EB ex de,h1
CE8B CD8DFE call FE8D CINT
CE8E EB ex de,h1
CE8F F1 pop af
CE90 C9 ret
```

```
***** aller chercher valeur 16 bits expression adresse
CE91 CDFBCE call CEFB aller chercher expression
CE94 F5 push af
CE95 C5 push bc
CE96 E5 push h1
CE97 CDC2FE call FEC2 UNT
CE9A EB ex de,h1
CE9B E1 pop h1
CE9C C1 pop bc
CE9D F1 pop af
CE9E C9 ret
```

```
***** aller chercher expression chaîne et paramètres
CE9F CDFBCE call CEFB aller chercher expression
CEA2 C3DAFB jp FBDA aller chercher paramètres chaîne
```


BASIC 1.0

```
***** aller chercher expression chaîne
CEA5 CDFBCE call CEFB aller chercher expression
CEA8 C33CFF jp FF3C Type 'chaîne', sinon 'Type mismatch'

CEAB 1E05 ld e,05 'Improper argument'
CEAD C394CA jp CA94 sortir message d'erreur

***** aller chercher zone de numéros de ligne
CEB0 010100 ld bc,0001 1
CEB3 11FFFF ld de,FFFF et 65535 par défaut
CEB6 CD55DD call DD55 tester si virgule
```

BASIC 1.0

CEB9	D451DD	call	nc,DD51	fin de l'instruction ?
CEBC	D8	ret	c	oui
CEBD	FE23	cp	23	'#' (numéro de canal) ?
CEBF	C8	ret	z	
CECO	FEF5	cp	F5	'-'
CEC2	280A	jr	z,CECE	
CEC4	CDE1CE	call	CEE1	aller chercher No de ligne dans de
CEC7	42	ld	b,d	
CEC8	4B	ld	c,e	copier dans bc
CEC9	C8	ret	z	
CECA	CD55DD	call	DD55	virgule suit ?
CECD	D8	ret	c	oui
CECE	CD37DD	call	DD37	Tester si encore un caractère
CED1	F5	db	F5	'-'
CED2	11FFFF	ld	de,FFFF	65535 comme valeur finale par défaut
CED5	C8	ret	z	
CED6	CD55DD	call	DD55	tester si virgule.
CED9	D8	ret	c	
CEDA	CDE1CE	call	CEE1	prendre numéro de ligne dans de
CEDD	C455DD	call	nz,DD55	tester si virgule
CEE0	C9	ret		

CEE1	7E	ld	a,(hl)	prendre numéro de ligne dans de
CEE2	23	inc	hl	type constante
CEE3	5E	ld	e,(hl)	
CEE4	23	inc	hl	valeur dans de
CEE5	56	ld	d,(hl)	
CEE6	FE1E	cp	1E	numéro de ligne ?
CEE8	280E	jr	z,CEF8	oui, terminé
CEEA	FE1D	cp	1D	adresse de ligne ?
CEEC	C27BDO	jp	nz,D07B	non, 'Syntax error'
CEEF	E5	push	hl	
CEFO	EB	ex	de,hl	hl désigne début de ligne
CEF1	23	inc	hl	
CEF2	23	inc	hl	
CEF3	23	inc	hl	
CEF4	5E	ld	e,(hl)	
CEF5	23	inc	hl	numéro de ligne dans de
CEF6	56	ld	d,(hl)	

BASIC 1.0

CEF7	E1	pop	hl	
CEF8	C33FDD	jp	DD3F	ignorer espaces
*****				aller chercher expression
CEFB	C5	push	bc	
CEFC	2B	dec	hl	
CEFD	0600	ld	b,00	code de hiérarchie
CEFF	CD07CF	call	CF07	aller chercher terme
CF02	C1	pop	bc	
CF03	2B	dec	hl	
CF04	C33FDD	jp	DD3F	ignorer espaces
*****				aller chercher terme
CF07	C5	push	bc	
CF08	CDCBCF	call	CFCB	aller chercher terme
CF0B	E5	push	hl	
CF0C	E1	pop	hl	
CF0D	C1	pop	bc	
CF0E	7E	ld	a,(hl)	
CF0F	FE44	cp	EE	'>'
CF11	D8	ret	c	inférieur ?
CF12	FEFE	cp	FE	'NOT'
CF14	D0	ret	nc	supérieur égal ?
CF15	FEF4	cp	F4	'+'
CF17	3840	jr	c,CF59	inférieur, alors opérateur de comparaison
CF19	CC45FF	call	z,FF45	'+', alors tester si chaîne
CF1C	2012	jr	nz,CF30	pas chaîne
CF1E	C5	push	bc	
CF1F	E5	push	hl	
CF20	2AC2B0	ld	hl,(BOC2)	Stringdescriptor
CF23	E3	ex	(sp),hl	sur pile
CF24	CDCBCF	call	CFCB	aller chercher terme suivant
CF27	CD3CFF	call	FF3C	Type 'chaîne', sinon 'Type mismatch'
CF2A	E3	ex	(sp),hl	
CF2B	CD63F8	call	F863	addition de chaînes
CF2E	18DC	jr	CF0C	traiter terme suivant
*****				opérateurs arithmétiques
CF30	7E	ld	a,(hl)	

BASIC 1.0

CF31	D6F4	sub	F4	moins F4
CF33	87	add	a,a	
CF34	87	add	a,a	fois 4
CF35	C681	add	a,81	
CF37	5F	ld	e,a	plus CF81, adresse de table
CF38	CECF	adc	a,CF	
CF3A	93	sub	e	
CF3B	57	ld	d,a	
CF3C	EB	ex	de,h1	
CF3D	78	ld	a,b	
CF3E	BE	cp	(h1)	
CF3F	EB	ex	de,h1	
CF40	D0	ret	nc	
CF41	C5	push	bc	
CF42	CD53FF	call	FF53	placer résultat sur la pile Basic
CF45	D5	push	de	
CF46	C5	push	bc	
CF47	1A	ld	a,(de)	code de hiérarchie
CF48	47	ld	b,a	
CF49	CD07CF	call	CF07	aller chercher terme
CF4C	C1	pop	bc	
CF4D	E3	ex	(sp),h1	
CF4E	23	inc	h1	
CF4F	EB	ex	de,h1	
CF50	79	ld	a,c	
CF51	CDA0F5	call	F5A0	libérer place dans la pile Basic
CF54	CDFBFF	call	FFFB	Jp (de), exécuter opération
CF57	18B3	Jr	CF0C	traiter terme suivant
*****				opérateurs de comparaison
CF59	78	ld	a,b	
CF5A	FE0A	cp	0A	
CF5C	D0	ret	nc	
CF5D	C5	push	bc	
CF5E	7E	ld	a,(h1)	Token
CF5F	D6ED	sub	ED	moins Offset
CF61	47	ld	b,a	
CF62	CD45FF	call	FF45	Tester si chaîne
CF65	11A9CF	ld	de,CFA9	Adresse pour comparaisons arithmétiques

BASIC 1.0

CF68	20D8	jr	nz,CF42	pas chaîne
CF6A	E5	push	hl	
CF6B	2AC2B0	ld	hl,(BOC2)	Stringdescriptor
CF6E	E3	ex	(sp),hl	sur pile
CF6F	C5	push	bc	
CF70	060A	ld	b,0A	code de hiérarchie
CF72	CD07CF	call	CF07	aller chercher terme
CF75	C1	pop	bc	
CF76	E3	ex	(sp),hl	
CF77	C5	push	bc	
CF78	CD97F8	call	F897	comparaison de chaînes
CF7B	C1	pop	bc	
CF7C	CDAFCF	call	CFAF	aller chercher résultat de comparaison
CF7F	188B	jr	CF0C	traiter terme suivant

***** codes de hiérarchie des opérateurs Basic + adresses

CF81	0C	db	0C	F4, '+'
CF82	C3CCFC	jp	FCCC	
CF85	0C	db	0C	F5, '-'
CF86	C3E1FC	jp	FCE1	
CF89	12	db	12	F6, '**'
CF8A	C3F5FC	jp	FCF5	
CF8D	12	db	12	F7, '/'
CF8E	C312FD	jp	FD12	
CF91	16	db	16	F8, '^'
CF92	C3F4D4	jp	D4F4	
CF95	10	db	10	F9, 'Backslash'
CF96	C337FD	jp	FD37	
CF99	06	db	06	FA, 'AND'
CF9A	C358FD	jp	FD58	
CF9D	0E	db	0E	FB, 'MOD'
CF9E	C349FD	jp	FD49	
CFA1	04	db	04	FC, 'OR'
CFA2	C363FD	jp	FD63	
CFA5	02	db	02	FD, 'XOR'
CFA6	C36DFD	jp	FD6D	

***** Comparaison arithmétique

CFA9	0A	ld	a,(bc)
------	----	----	--------

BASIC 1.0

CFAA	C5	push	bc	
CFAB	CD09FD	call	FD09	comparaison arithmétique
CFAE	C1	pop	bc	
CFAF	C601	add	a,01	
CFB1	8F	adc	a,a	
CFB2	A0	and	b	
CFB3	C6FF	add	a,FF	
CFB5	9F	sbc	a,a	
CFB6	C305FF	jp	FF05	accepter signe
*****				'-' signe négatif
CFB9	2B	dec	hl	
CFBA	0614	ld	b,14	code de hiérarchie
CFBC	CD07CF	call	CF07	aller chercher terme
CFBF	C389FD	jp	FD89	changer signe
*****				opérateur Basic NOT
CFC2	2B	dec	hl	
CFC3	0608	ld	b,08	code de hiérarchie
CFC5	CD07CF	call	CF07	aller chercher terme
CFC8	C377FD	jp	FD77	opérateur NOT
*****				aller chercher expression
CFCB	CD3FDD	call	DD3F	ignorer espaces
*****				aller chercher expression
CFCE	281D	jr	z,CFED	'Operand missing'
CFD0	FE0E	cp	0E	
CFD2	3839	jr	c,D00D	aller chercher variable
CFD4	FE20	cp	20	
CFD6	3854	jr	c,D02C	aller chercher valeur numérique
CFD8	FE22	cp	22	'''
CFDA	CACBF7	jp	z,F7CB	au traitement de la chaîne
CFDD	FEFF	cp	FF	fonction ?
CFDF	CA80D0	jp	z,D080	au calcul de fonction
CFE2	E5	push	hl	
CFE3	21F2CF	ld	hl,CFF2	adresse de base de la table
CFE6	CD93FF	call	FF93	rechercher dans la table
CFE9	E3	ex	(sp),hl	

BASIC 1.0

CFEA	C33FDD	jp	DD3F	ignorer espaces
CFED	1E16	ld	e,16	'Operand missing'
CFEF	C394CA	jp	CA94	sortir message d'erreur

fonctions spéciales

CFF2	08	db	08	nombre d'entrées dans la table
CFF3	78D0	dw	D078	pas trouvé, 'Syntax error'
CFF5	F5	db	F5	'-'
CFF6	B9CF	dw	CFB9	
CFF8	F4	dw	F4	'+'
CFF9	CECF	dw	CFCE	
CFFB	28	db	28	'('
CFFC	70D0	dw	D070	
CFFE	FE	db	FE	'NOT'
CFFF	C2CF	dw	CFC2	
D001	E3	db	E3	'ERL'
D002	EED0	dw	D0EE	
D004	E4	db	E4	'FN'
D005	30D1	dw	D130	
D007	AC	db	AC	'MID\$'
D008	4BF9	dw	F94B	
D00A	40	db	40	'4j'
D00B	FAD0	dw	D0FA	

aller chercher variable

D00D	CD90D6	call	D690	aller chercher adresse de variable
D010	300B	jr	nc,D01D	pas encore initialisée ?
D012	FE03	cp	03	type de variable
D014	280F	jr	z,D025	chaîne ?
D016	E5	push	hl	
D017	EB	ex	de,hl	
D018	CD4BFF	call	FF4B	
D01B	E1	pop	hl	
D01C	C9	ret		
D01D	FE03	cp	03	chaîne ?
D01F	C2F3FE	jp	nz,FEF3	supprimer variable
D022	112BD0	ld	de,D02B	fixer valeur à zéro
D025	EB	ex	de,hl	

BASIC 1.0

D026 22C2B0 ld (BOC2),hl
 D029 EB ex de,hl
 D02A C9 ret

D02B 00 db 00 zéro

aller chercher valeur numérique
 D02C D60E sub 0E ôter offset
 D02E FE0A cp 0A inférieur 10 ?
 D030 381D jr c,D04F oui, aller chercher chiffre
 D032 23 inc hl
 D033 FE0B cp 0B valeur sur un octet ?
 D035 2817 jr z,D04E
 D037 FE0F cp 0F valeur sur deux octets (déc, hex, bin) ?
 D039 380E jr c,D049
 D03B FE11 cp 11
 D03D 381A jr c,D059 valeur à virgule flottante ?
 D03F 203A jr nz,D07B 'Syntax error'
 D041 3E05 ld a,05 'Real'
 D043 CD4BFF call FF4B
 D046 2B dec hl
 D047 1824 jr D06D ignorer espaces

aller chercher valeur deux octets
 D049 5E ld e,(hl)
 D04A 23 inc hl
 D04B 56 ld d,(hl)
 D04C 1804 jr D052 accepter nombre entier dans de

aller chercher valeur sur un octet
 D04E 7E ld a,(hl)
 D04F 5F ld e,a
 D050 1600 ld d,00 fixer octet fort sur zéro
 D052 EB ex de,hl
 D053 CD0DFF call FF0D accepter nombre entier dans hl
 D056 EB ex de,hl
 D057 1814 jr D06D ignorer espaces

BASIC 1.0

***** aller chercher valeur à virgule flottante

```

D059 5E      ld      e,(h1)
D05A 23      inc     hl
D05B 56      ld      d,(hl)
D05C E5      push   hl
D05D FE0F    cp      0F
D05F 2007    jr      nz,D068
D061 13      inc     de
D062 EB      ex     de,hl
D063 23      inc     hl
D064 23      inc     hl
D065 5E      ld      e,(hl)
D066 23      inc     hl
D067 56      ld      d,(hl)
D068 EB      ex     de,hl
D069 CD60FE  call   FE60      type de variable sur 'Real'
D06C E1      pop    hl
D06D C33FDD  jp     DD3F      ignorer espaces

```

***** '(' aller chercher valeur entre parenthèses

```

D070 CDFBCE  call   CEFB      aller chercher expression
D073 CD37DD  call   DD37      tester si encore un caractère
D076 29      db     29        ')'
D077 C9      ret

```

```

D078 CD0DAC  call   AC0D      ret
D07B 1E02    ld     e,02      'Syntax error'
D07D C394CA  jp     CA94      sortir message d'erreur

```

```

D080 23      inc     hl      calcul de fonction
D081 4E      ld     c,(hl)  augmenter pointeur de programme
D082 CD3FDD  call   DD3F      aller chercher token
D085 79      ld     a,c      ignorer espaces
D086 FE40    cp     40      tester token
D088 3805    jr     c,D08F   40 - 48, variable réservée
D08A FE49    cp     49
D08C DABBDO  jp     c,DOBB
D08F CD37DD  call   DD37      tester si encore un caractère

```

BASIC 1.0

```

D092 28      db      28      '(('
D093 79      ld      a,c
D094 87      add     a,a
D095 C61E    add     a,1E
D097 4F      ld      c,a
D098 FE59    cp      59
D09A 300D    jr      nc,DOA9    'Syntax error'
D09C FE1D    cp      1D
D09E 380E    jr      c,DOAE    calculer fonction
DOA0 CD70D0  call     D070    aller chercher argument de la
                                fonction entre parenthèses

DOA3 E5      push    h1
DOA4 CDAED0  call     DOAE    calculer fonction
DOA7 E1      pop     h1
DOA8 C9      ret

DOA9 CDOAAC  call     ACOA    ret
DOAC 18CD    jr      D07B    'Syntax error'

DOAE E5      push    h1
DOAF 0600    ld      b,00
DOB1 2190D1  ld      h1,D190  Adresses des fonctions
DOB4 09      add     h1,bc
DOB5 7E      ld      a,(h1)
DOB6 23      inc     h1
DOB7 66      ld      h,(h1)
DOB8 6F      ld      l,a
DOB9 E3      ex     (sp),h1
DOBA C9      ret

DOBB E5      push    h1
DOBC 4F      ld      c,a
DOBD 0600    ld      b,00
DOBF 214AD0  ld      h1,D04A
DOC2 09      add     h1,bc
DOC3 09      add     h1,bc
DOC4 7E      ld      a,(h1)
DOC5 23      inc     h1
DOC6 66      ld      h,(h1)
DOC7 6F      ld      l,a

```

BASIC 1.0

DOC8 E3 ex (sp),h1
 DOC9 C9 ret

***** Adresses des variables réservées

DOCA 17C4 dw C417 40, EOF
 DOCC DCDO dw DODC 41, ERR
 DOCE F4D0 dw DOF4 42, HIMEM
 DODO 24FA dw FA24 43, INKEY\$
 DOD2 DBD4 dw D4DB 44, PI
 DOD4 84D5 dw D584 45, RND
 DOD6 E5D0 dw DOE5 46, TIME
 DOD8 07D1 dw D107 47, XPOS
 DODA 0ED1 dw D10E 48, YPOS

***** ERR

DODC E5 push h1
 DODD 3AAAAD ld a,(ADAA) numéro ERROR
 DOE0 CDOAFF call FFOA accepter contenu accu comme nombre entier
 DOE3 E1 pop h1
 DOE4 C9 ret

***** TIME

DOE5 E5 push h1
 DOE6 CD0BD call BDOD KL TIME PLEASE
 DOE9 CD7CFE call FE7C convertir valeur 4 octets en valeur à virgule flottante
 DOE0 E1 pop h1
 DOE1 C9 ret

***** ERL

DOEE E5 push h1
 DOEF CDDFCA call CADF aller chercher numéro de ligne ERROR
 DOF2 180E jr D102

***** HIMEM

DOF4 E5 push h1
 DOF5 2A7BAE ld h1,(AE7B) HIMEM
 DOF8 1808 jr D102

BASIC 1.0

```

*****
DOFA CD90D6 call D690 '4j', pointeur de variable
DOFD D2ABCE jp nc,CEAB aller chercher adresse de variable
D100 E5 push hl absent, 'Improper argument'
D101 EB ex de,hl
D102 CD60FE call FE60 accepter valeur
D105 E1 pop hl
D106 C9 ret

***** XPOS
D107 E5 push hl
D108 CDC6BB call - BBC6 GRA ASK CURSOR
D10B EB ex de,hl
D10C 1804 jr D112

***** YPOS
D10E E5 push hl
D10F CDC6BB call BBC6 GRA ASK CURSOR
D112 CDODFF call FF0D accepter nombre entier dans hl
D115 E1 pop hl
D116 C9 ret

***** instruction Basic DEF
D117 CD37DD call DD37 tester si encore un caractère
D11A E4 db E4 'FN'
D11B EB ex de,hl
D11C CDD6DD call DDD6 prendre numéro de ligne dans hl
D11F EB ex de,hl
D120 1E0C ld e,0C 'Invalid direct command'
D122 D294CA jp nc,CA94 mode direct, sortir message d'erreur

D125 CDA2D6 call D6A2 chercher fonction
D128 EB ex de,hl
D129 73 ld (hl),e
D12A 23 inc hl
D12B 72 ld (hl),d
D12C EB ex de,hl
D12D C3EFE8 jp E8EF ignorer reste de l'instruction

***** fonction Basic FN

```

BASIC 1.0

D130	CDA2D6	call	D6A2	chercher fonction
D133	C5	push	bc	
D134	E5	push	h1	
D135	EB	ex	de,h1	
D136	5E	ld	e,(h1)	
D137	23	inc	h1	
D138	56	ld	d,(h1)	
D139	EB	ex	de,h1	
D13A	7C	ld	a,h	
D13B	B5	or	l	
D13C	1E12	ld	e,12	'Unknown user function'
D13E	CA94CA	jp	z,CA94	sortir message d'erreur
D141	CD07DA	call	DA07	
D144	7E	ld	a,(h1)	
D145	FE28	cp	28	'('
D147	202C	Jr	nz,D175	
D149	CD3FDD	call	DD3F	ignorer espaces
D14C	E3	ex	(sp),h1	
D14D	CD37DD	call	DD37	tester si encore un caractère
D150	28	db	28	'('
D151	E3	ex	(sp),h1	
D152	CD4BDA	call	DA4B	
D155	E3	ex	(sp),h1	
D156	D5	push	de	
D157	CDFBCE	call	CEFB	aller chercher expression
D15A	E3	ex	(sp),h1	
D15B	78	ld	a,b	
D15C	CD66D6	call	D666	
D15F	E1	pop	h1	
D160	CD55DD	call	DD55	virgule suit ?
D163	3007	Jr	nc,D16C	non
D165	E3	ex	(sp),h1	
D166	CD37DD	call	DD37	tester si encore un caractère
D169	2C	db	2C	','
D16A	18E6	Jr	D152	
D16C	CD37DD	call	DD37	tester si encore un caractère
D16F	29	db	29	')
D170	E3	ex	(sp),h1	

BASIC 1.0

D171	CD37DD	call	DD37	tester si encore un caractère
D174	29	db	29	')
D175	CD27DA	call	DA27	
D178	CD37DD	call	DD37	tester si encore un caractère
D17B	EF	db	EF	'='
D17C	CDFBCE	call	CEFB	aller chercher expression
D17F	C27BD0	jp	nz,D07B	'Syntax error'
D182	CD30DA	call	DA30	
D185	CD45FF	call	FF45	Tester si chaîne
D188	CC49FB	call	z,FB49	oui
D18B	E1	pop	h1	
D18C	F1	pop	af	
D18D	C3D7FE	jp	FED7	

***** fonctions Basic à plusieurs arguments

D190	BAF8	dw	F8BA	71, BIN\$
D192	EAF8	dw	F8EA	72, DEC\$
D194	C4F8	dw	F8C4	73, HEX\$
D196	A1FA	dw	FAA1	74, INSTR
D198	3CF9	dw	F93C	75, LEFT\$
D19A	EED1	dw	D1EE	76, MAX
D19C	EAD1	dw	D1EA	77, MIN
D19E	76C2	dw	C276	78, POS
D1A0	43F9	dw	F943	79, RIGHT\$
D1A2	19D2	dw	D219	7A, ROUND
D1A4	36FA	dw	FA36	7B, STRING\$
D1A6	E9C4	dw	C4E9	7C, TEST
D1A8	EEC4	dw	C4EE	7D, TESTR
D1AA	ABCE	dw	CEAB	7E, 'Improper argument'
D1AC	62C2	dw	C262	7F, VPOS

***** Adresses des fonctions Basic

D1AE	85FD	dw	FD85	00, ABS
D1B0	10FA	dw	FA10	01, ASC
D1B2	3ED5	dw	D53E	02, ATN
D1B4	16FA	dw	FA16	03, CHR\$
D1B6	8DFE	dw	FE8D	04, CINT
D1B8	34D5	dw	D534	05, COS
D1BA	ECFE	dw	FEEC	06, CREAL
D1BC	20D5	dw	D520	07, EXP

BASIC 1.0

D1BE	E8FD	dw	FDE8	08, FIX
D1C0	2DFC	dw	FC2D	09, FRE
D1C2	09D4	dw	D409	0A, INKEY
D1C4	6DF1	dw	F16D	0B, INP
D1C6	EDFD	dw	FDED	0C, INT
D1C8	23D4	dw	D423	0D, JOY
D1CA	0AFA	dw	FA0A	0E, LEN
D1CC	2AD5	dw	D52A	0F, LOG
D1CE	25D5	dw	D525	10, LOG10
D1D0	34F8	dw	F834	11, LOWER\$
D1D2	58F1	dw	F158	12, PEEK
D1D4	9FC9	dw	C99F	13, REMAIN
D1D6	02FF	dw	FF02	14, SGN
D1D8	2FD5	dw	D52F	15, SIN
D1DA	57FA	dw	FA57	16, SPACE\$
D1DC	29D3	dw	D329	17, SQ
D1DE	EFD4	dw	D4EF	18, SQR
D1E0	1EF9	dw	F91E	19, STR\$
D1E2	39D5	dw	D539	1A, TAN
D1E4	C2FE	dw	FEC2	1B, UNT
D1E6	42F8	dw	F842	1C, UPPER\$
D1E8	77FA	dw	FA77	1D, VAL

BASIC 1.0

```

*****
D1EA 06FF ld b,FF fonction Basic MIN
D1EC 1802 jr D1F0 Flag pour MIN

*****
D1EE 0601 ld b,01 fonction Basic MAX
D1F0 CDFBCE call CEFB Flag pour MAX
D1F3 CD55DD call DD55 aller chercher expression
D1F6 301C jr nc,D214 virgule suit ?
D1F8 CD53FF call FF53 non, terminé
D1FB CDFBCE call CEFB placer variable sur la pile Basic
D1FE E5 push hl aller chercher expression
D1FF 79 ld a,c
D200 CDA0F5 call F5A0 libérer place dans la pile Basic
D203 C5 push bc
D204 E5 push hl
D205 CD09FD call FD09 comparaison arithmétique
D208 E1 pop hl
D209 C1 pop bc
D20A B7 or a
D20B 2804 jr z,D211
D20D B8 cp b
D20E C44EFF call nz,FF4E aller chercher résultat de la
comparaison

D211 E1 pop hl
D212 18DF jr D1F3 argument suivant

D214 CD37DD call DD37 tester si encore un caractère
D217 29 db 29 ')'
D218 C9 ret

*****
D219 CDFBCE call CEFB fonction Basic ROUND
D21C CD53FF call FF53 aller chercher expression
D21F CD55DD call DD55 et placer sur la pile Basic
D222 110000 ld de,0000 virgule suit ?
D225 DC86CE call c,CE86 valeur par défaut 0
oui, aller chercher valeur entière
avec signe

D228 CD37DD call DD37 tester si encore un caractère
D22B 29 db 29 ')'

```


BASIC 1.0

D22C	E5	push	hl	
D22D	D5	push	de	
D22E	212700	ld	hl,0027	39
D231	19	add	hl,de	additionner
D232	114F00	ld	de,004F	79
D235	CDB8FF	call	FFB8	comparer hl <> de
D238	D2ABCE	jp	nc,CEAB	supérieur, 'Improper argument'
D23B	D1	pop	de	
D23C	79	ld	a,c	
D23D	CDA0F5	call	F5A0	libérer place dans la pile Basic
D240	43	ld	b,e	nombre d'arrondissement dans b
D241	CDAFFD	call	FFDAF	arrondir le nombre
D244	E1	pop	hl	
D245	C9	ret		

***** instruction Basic CAT

D246	C0	ret	nz	
D247	E5	push	hl	
D248	CDADD2	call	D2AD	interrompre I/O cassette
D24B	CD37F6	call	F637	mettre en place buffer de sortie
D24E	CD9BBC	call	BC9B	CAS CATALOG
D251	CD71F6	call	F671	libérer buffer de sortie
D254	E1	pop	hl	
D255	C9	ret		

***** instruction Basic OPENOUT

D256	CD73D2	call	D273	aller chercher nom de fichier
D259	CD37F6	call	F637	organiser buffer de sortie
D25C	C38CBC	jp	BC8C	CAS OUT OPEN

***** instruction Basic OPENIN

D25F	CD6AD2	call	D26A	aller chercher nom, ouvrir fichier
D262	FE16	cp	16	fichier ASCII ?
D264	C8	ret	z	
D265	1E19	ld	e,19	'File type error'
D267	C394CA	jp	CA94	sortir message d'erreur

***** aller chercher nom, ouvrir fichier d'entrée

D26A	CD73D2	call	D273	aller chercher nom de fichier
D26D	CD32F6	call	F632	mettre en place buffer d'entrée

BASIC 1.0

```

D270 C377BC   jp     BC77         CAS IN OPEN

*****
D273 CD9FCE   call   CE9F         aller chercher paramètres et
                        expression de chaîne

D276 E3       ex     (sp),hl
D277 EB       ex     de,hl
D278 CD85D2   call   D285         tester si messages du système
D27B CA6BCB   jp     z,CB6B       interruption par 'ESC'
D27E E1       pop    hl
D27F D8       ret    c
D280 1E1B     ld     e,1B         'File already open'
D282 C394CA   jp     CA94         sortir message d'erreur

*****
D285 D5       push   de
D286 0E00     ld     c,00         sortir flag pour messages
D288 78       ld     a,b         longueur du nom de fichier
D289 B7       or     a
D28A 2808     jr     z,D294       pas de nom de fichier ?
D28C 7E       ld     a,(hl)       premier caractère
D28D FE21     cp     21           '!'
D28F 2003     jr     nz,D294      non
D291 23       inc    hl           fixer pointeur sur second caractère
D292 05       dec    b           diminuer longueur
D293 0D       dec    c           interdire flag pour messages
D294 79       ld     a,c
D295 C36BBC   jp     BC6B         CAS NOISY

***** instruction Basic CLOSEIN
D298 E5       push   hl
D299 CD7ABC   call   BC7A         CAS IN CLOSE
D29C CD6DF6   call   F66D         libérer buffer d'entrée
D29F E1       pop    hl
D2A0 C9       ret

***** instruction Basic CLOSEOUT
D2A1 E5       push   hl
D2A2 CD8FBC   call   BC8F         CAS OUT CLOSE
D2A5 CA6BCB   jp     z,CB6B       'Break in numéro de ligne'

```

BASIC 1.0

```
D2A8 CD71F6 call F671 libérer buffer de sortie
D2AB E1 pop hl
D2AC C9 ret
```

***** interromptre I/O cassette

```
D2AD C5 push bc
D2AE D5 push de
D2AF E5 push hl
D2B0 CD7DBC call BC7D CAS IN ABANDON
D2B3 CD6DF6 call F66D libérer buffer d'entrée
D2B6 CD92BC call BC92 CAS OUT ABANDON
D2B9 CD71F6 call F671 libérer buffer de sortie
D2BC E1 pop hl
D2BD D1 pop de
D2BE C1 pop bc
D2BF C9 ret
```

***** instruction Basic SOUND

```
D2C0 CD67CE call CE67 aller chercher valeur 8 bits
D2C3 32B2AD ld (ADB2),a état canal
D2C6 CD37DD call DD37 tester si encore un caractère
D2C9 2C db 2C ','
D2CA CDFFD3 call D3FF aller chercher argument 0 à 4095
D2CD ED53B5AD ld (ADB5),de période du ton
D2D1 CD55DD call DD55 virgule suit ?
D2D4 111400 ld de,0014 valeur par défaut 20
D2D7 DC86CE call c,CE86 oui, aller chercher valeur entière
avec signe
D2DA ED53B9AD ld (ADB9),de Durée
D2DE 010C10 ld bc,100C max. 15, défaut 12
D2E1 CD0DD3 call D30D aller chercher argument s'il y en a
un
D2E4 32B8AD ld (ADB8),a volume
D2E7 0E00 ld c,00 max. 15, défaut 0
D2E9 CD0DD3 call D30D aller chercher argument s'il y en a
un
D2EC 32B3AD ld (ADB3),a courbe d'enveloppe de volume
D2EF CD0DD3 call D30D aller chercher argument s'il y en a
un
D2F2 32B4AD ld (ADB4),a Courbe d'enveloppe du ton
```

BASIC 1.0

D2F5	0620	ld	b,20	max. 31, défaut 0
D2F7	CD0DD3	call	D30D	aller chercher argument s'il y en a un
D2FA	32B7AD	ld	(ADB7),a	période bruit
D2FD	CD4ADD	call	DD4A	fin de l'instruction, sinon 'Syntax error'
D300	E5	push	hl	
D301	21B2AD	ld	hl,ADB2	Adesse bloc paramètres Sound
D304	CDAABC	call	BCAA	SOUND QUEUE
D307	E1	pop	hl	
D308	D8	ret	c	
D309	F1	pop	af	
D30A	C371DD	jp	DD71	à la boucle de l'interpréteur

*****				aller chercher valeur 8 bits s'il y en a une
D30D	CD55DD	call	DD55	virgule suit ?
D310	79	ld	a,c	charger valeur défaut
D311	D0	ret	nc	pas virgule, terminé
D312	7E	ld	a,(hl)	
D313	FE2C	cp	2C	','
D315	79	ld	a,c	
D316	C8	ret	z	
D317	CD67CE	call	CE67	aller chercher valeur 8 bits holen supérieur égal b ?
D31A	B8	cp	b	non
D31B	D8	ret	c	
D31C	182B	Jr	D349	'Improper argument'

*****				instruction Basic RELEASE
D31E	0608	ld	b,08	8
D320	CD17D3	call	D317	aller chercher valeur 8 bits < 8
D323	E5	push	hl	
D324	CDB3BC	call	BCB3	SOUND RELEASE
D327	E1	pop	hl	
D328	C9	ret		

*****				fonction Basic SQ
D329	CD8DFE	call	FE8D	CINT
D32C	7D	ld	a,l	
D32D	B7	or	a	

BASIC 1.0

D32E	1F	rra		
D32F	3806	jr	c,D337	
D331	1F	rra		
D332	3803	jr	c,D337	
D334	1F	rra		
D335	3012	jr	nc,D349	'Improper argument'
D337	B4	or	h	
D338	200F	jr	nz,D349	'Improper argument'
D33A	7D	ld	a,l	
D33B	CDADBC	call	BCAD	SOUND CHECK
D33E	C30AFF	jp	FF0A	accepter contenu accu comme nombre entier

 D341 CD86CE call CE86 aller chercher argument -128 à +127
 aller chercher valeur entière avec signe

D344	7B	ld	a,e	
D345	87	add	a,a	
D346	9F	sbc	a,a	
D347	BA	cp	d	
D348	C8	ret	z	
D349	1E05	ld	e,05	'Improper argument'
D34B	C394CA	jp	CA94	sortir message d'erreur

 D34E CD6DCE call CE6D instruction Basic ENV
 aller chercher valeur 8 bits non nulle
 supérieur égal 16 ?
 'Improper argument'

D351	FE10	cp	10	
D353	30F4	jr	nc,D349	'Improper argument'
D355	F5	push	af	
D356	1167D3	ld	de,D367	
D359	CDD8D3	call	D3D8	
D35C	F1	pop	af	
D35D	E5	push	hl	
D35E	21BBAD	ld	hl,ADBB	
D361	71	ld	(hl),c	
D362	CDBCBC	call	BCBC	SOUND AMPL ENVELOPE
D365	E1	pop	hl	
D366	C9	ret		

BASIC 1.0

D367	7E	ld	a,(hl)	
D368	FEEF	cp	EF	'='
D36A	2012	jr	nz,D37E	
D36C	CD3FDD	call	DD3F	ignorer espaces
D36F	0610	ld	b,10	16
D371	CD17D3	call	D317	aller chercher valeur 8 bits < 16
D374	F680	or	80	mettre bit 7
D376	4F	ld	c,a	
D377	CD37DD	call	DD37	tester si encore un caractère
D37A	2C	db	2C	','
D37B	C391CE	jp	CE91	aller chercher valeur 16 bits
D37E	0680	ld	b,80	128
D380	CD17D3	call	D317	aller chercher valeur 8 bits < 128
D383	1840	jr	D3C5	,
*****			instruction Basic ENT	
D385	CD41D3	call	D341	aller chercher argument -128 à +127
D388	7A	ld	a,d	
D389	B7	or	a	
D38A	7B	ld	a,e	
D38B	2802	jr	z,D38F	zéro ?
D38D	2F	cpl	a	
D38E	3C	inc	a	
D38F	5F	ld	e,a	
D390	B7	or	a	zéro ?
D391	28B6	jr	z,D349	'Improper argument'
D393	FE10	cp	10	supérieur égal 16 ?
D395	30B2	jr	nc,D349	'Improper argument'
D397	D5	push	de	
D398	11AED3	ld	de,D3AE	
D39B	CDD8D3	call	D3D8	
D39E	D1	pop	de	
D39F	E5	push	hl	
D3A0	21BBAD	ld	hl,ADBB	
D3A3	7A	ld	a,d	
D3A4	E680	and	80	
D3A6	B1	or	c	
D3A7	77	ld	(hl),a	
D3A8	7B	ld	a,e	

BASIC 1.0

```
D3A9 CDBFBC call BCBF SOUND TONE ENVELOPE
D3AC E1 pop hl
D3AD C9 ret
```

```
D3AE 7E ld a,(hl)
D3AF FEFF cp EF '='
D3B1 200D jr nz,D3C0
D3B3 CD3FDD call DD3F ignorer espaces
D3B6 CDFFD3 call D3FF aller chercher valeur de 0 à 4095
D3B9 7A ld a,d
D3BA C6F0 add a,F0
D3BC 4F ld c,a
D3BD 43 ld b,e
D3BE 180E jr D3CE

D3C0 06F0 ld b,F0 240
D3C2 CD17D3 call D317 aller chercher valeur 8 bits < 240
D3C5 4F ld c,a
D3C6 CD37DD call DD37 tester si encore un caractère
D3C9 2C db 2C ','
D3CA CD41D3 call D341 aller chercher argument -128 à +127
D3CD 43 ld b,e
D3CE CD37DD call DD37 tester si encore un caractère
D3D1 2C db 2C ','
D3D2 CD67CE call CE67 aller chercher valeur 8 bits
D3D5 57 ld d,a
D3D6 58 ld e,b
D3D7 C9 ret

D3D8 010005 ld bc,0500
D3DB CD55DD call DD55 tester si virgule
D3DE 301C jr nc,D3FC
D3E0 D5 push de
D3E1 C5 push bc
D3E2 CDFBFF call FFFB jp (de)
D3E5 79 ld a,c
D3E6 C1 pop bc
D3E7 C5 push bc
D3E8 E5 push hl
```

BASIC 1.0

D3E9	21BCAD	ld	hl,ADBC	
D3EC	0600	ld	b,00	
D3EE	09	add	hl,bc	
D3EF	09	add	hl,bc	
D3F0	09	add	hl,bc	
D3F1	77	ld	(hl),a	
D3F2	23	inc	hl	
D3F3	73	ld	(hl),e	
D3F4	23	inc	hl	
D3F5	72	ld	(hl),d	
D3F6	E1	pop	hl	
D3F7	C1	pop	bc	
D3F8	0C	inc	c	
D3F9	D1	pop	de	
D3FA	10DF	djnz	D3DB	
D3FC	C34ADD	jp	DD4A	fin de l'instruction, sinon 'Syntax error'

*****				aller chercher argument 0 à 4095
D3FF	CD86CE	call	CE86	aller chercher valeur entière avec signe
D402	7A	ld	a,d	Hi-Byte
D403	E6F0	and	F0	Bits 12-15 mis ?
D405	C249D3	jp	nz,D349	oui, 'Improper argument'
D408	C9	ret		

*****				fonction Basic INKEY
D409	CD8DFE	call	FE8D	CINT
D40C	115000	ld	de,0050	80
D40F	CDB8FF	call	FFB8	comparer hl <> de
D412	3022	jr	nc,D436	'Improper argument'
D414	7D	ld	a,l	
D415	CD1EBB	call	BB1E	KM TEST KEY
D418	21FFFF	ld	hl,FFFF	-1, si pas enfoncée
D41B	2803	jr	z,D420	
D41D	69	ld	l,c	
D41E	2600	ld	h,00	
D420	C30DFF	jp	FF0D	accepter nombre entier dans hl

*****				fonction Basic JOY
-------	--	--	--	--------------------

BASIC 1.0

```

D423 CD24BB call BB24 KM GET JOYSTICK
D426 EB ex de,h1
D427 CD8DFE call FE8D CINT
D42A 7C ld a,h
D42B B5 or l
D42C 2802 jr z,D430
D42E 53 ld d,e
D42F 2B dec hl
D430 7C ld a,h
D431 B5 or l
D432 7A ld a,d
D433 CA0AFF jp z,FF0A accepter contenu accu comme nombre
entier
D436 C349D3 jp D349 'Improper argument'

```

instruction Basic KEY

```

D439 FE8D cp 8D 'DEF'
D43B 2819 jr z,D456
D43D 3E20 ld a,20
D43F CD17D3 call D317 aller chercher argument, valeur 8
bits
D442 F5 push af ranger numéro de touche
D443 CD37DD call DD37 tester si encore un caractère
D446 2C db 2C ','
D447 CD9FCE call CE9F aller chercher expression et
paramètres de chaîne
D44A 48 ld c,b longueur de chaîne dans c
D44B F1 pop af
D44C 47 ld b,a numéro de touche dans b
D44D E5 push hl
D44E EB ex de,h1 adresse de chaîne dans hl
D44F CD0FBB call BBOF KM SET EXPAND
D452 E1 pop hl
D453 30E1 jr nc,D436 'Improper argument'
D455 C9 ret

```

KEY DEF

```

D456 CD3FDD call DD3F ignorer espaces
D459 CD67CE call CE67 aller chercher valeur 8 bits,
Tastennnummer

```

BASIC 1.0

D45C	4F	ld	c,a	
D45D	FE50	cp	50	80
D45F	30D5	jr	nc,D436	supérieur égal, 'Improper argument'
D461	CD37DD	call	DD37	tester si encore un caractère
D464	2C	db	2C	','
D465	0602	ld	b,02	2
D467	CD17D3	call	D317	aller chercher argument < 2
D46A	1F	rra		
D46B	9F	sbc	a,a	
D46C	47	ld	b,a	
D46D	C5	push	bc	
D46E	E5	push	hl	
D46F	79	ld	a,c	
D470	CD39BB	call	BB39	KM SET REPEAT
D473	E1	pop	hl	
D474	C1	pop	bc	
D475	1127BB	ld	de,BB27	KM SET TRANSLATE
D478	CD84D4	call	D484	tester si encore un argument
D47B	112DBB	ld	de,BB2D	KM SET SHIFT
D47E	CD84D4	call	D484	tester si encore un argument
D481	1133BB	ld	de,BB33	KM SET CONTROL
D484	CD55DD	call	DD55	virgule suit ?
D487	D0	ret	nc	non, terminé
D488	D5	push	de	
D489	CD67CE	call	CE67	aller chercher valeur 8 bits
D48C	47	ld	b,a	
D48D	E3	ex	(sp),hl	
D48E	79	ld	a,c	
D48F	CD8FF	call	FFF8	jp (hl)
D492	E1	pop	hl	
D493	C9	ret		

				instruction Basic SPEED
D494	FEA4	cp	A4	'KEY'
D496	013FBB	ld	bc,BB3F	KM SET DELAY
D499	2810	jr	z,D4AB	
D49B	FEA2	cp	A2	'INK'
D49D	013EBC	ld	bc,BC3E	SCR SET FLASHING
D4A0	2809	jr	z,D4AB	
D4A2	FED9	cp	D9	'WRITE'

BASIC 1.0

```
D4A4 281D jr z,D4C3
D4A6 1E02 ld e,02 'Syntax error'
D4A8 C394CA jp CA94 sortir message d'erreur
```

SPEED KEY & INK

```
D4AB C5 push bc
D4AC CD3FDD call DD3F ignorer espaces
D4AF CD6DCE call CE6D aller chercher valeur 8 bits non
nulle
D4B2 4F ld c,a
D4B3 CD37DD call DD37 tester si encore un caractère
D4B6 2C db 2C ','
D4B7 CD6DCE call CE6D aller chercher valeur 8 bits non
nulle
D4BA 5F ld e,a
D4BB 51 ld d,c
D4BC C1 pop bc
D4BD EB ex de,hl
D4BE CDF9FF call FFF9 jp (bc)
D4C1 EB ex de,hl
D4C2 C9 ret
```

SPEED WRITE

```
D4C3 CD3FDD call DD3F ignorer espaces
D4C6 0602 ld b,02
D4C8 CD17D3 call D317 aller chercher argument < 2
D4CB E5 push hl
D4CC 21A700 ld hl,00A7 167
D4CF 3D dec a
D4D0 3E32 ld a,32
D4D2 2802 jr z,D4D6 zéro ?
D4D4 29 add hl,hl non, doubler constante de temps
D4D5 0F rrca
D4D6 CD68BC call BC68 CAS SET SPEED
D4D9 E1 pop hl
D4DA C9 ret
```

PI

```
D4DB E5 push hl
D4DC CD19FF call FF19 fixer type sur 'Real'
```

BASIC 1.0

D4DF	CD1DFF	call	FF1D	type de variable dans c, hl sur variable
D4E2	CD76BD	call	BD76	aller chercher 4S
D4E5	E1	pop	hl	
D4E6	C9	ret		
*****				instruction Basic DEG
D4E7	3EFF	ld	a,FF	FF = DEG
D4E9	1801	jr	D4EC	
*****				instruction Basic RAD
D4EB	AF	xor -	a	0 = RAD
D4EC	C373BD	jp	BD73	fixer mode DEG/RAD
*****				fonction Basic SQR
D4EF	0179BD	ld	bc,BD79	fonction SQR
D4F2	1816	jr	D50A	
*****				opérateur Basic '^'
D4F4	E5	push	hl	
D4F5	C5	push	bc	
D4F6	CDECFE	call	FEEC	CREAL
D4F9	EB	ex	de,hl	
D4FA	21CBAD	ld	hl,ADCB	mémoire provisoire pour nombre à virgule flottante
D4FD	CD3DBD	call	BD3D	copier variable de (de) dans (hl)
D500	C1	pop	bc	
D501	E3	ex	(sp),hl	
D502	79	ld	a,c	
D503	CD4BFF	call	FF4B	
D506	D1	pop	de	
D507	017CBD	ld	bc,BD7C	élévation à la puissance
D50A	CD19D5	call	D519	exécuter fonction
D50D	D8	ret	c	sans erreur ?
D50E	CAEACA	jp	z,CAEA	'Division by zero'
D511	FAF3CA	jp	m,CAF3	'Overflow'
D514	1E05	ld	e,05	'Improper argument'
D516	C394CA	jp	CA94	sortir message d'erreur

BASIC 1.0

D519	C5	push	bc	
D51A	D5	push	de	
D51B	CDECFE	call	FEEC	CREAL
D51E	D1	pop	de	

BASIC 1.0

```

D51F C9      ret      exécuter fonction

*****
D520 0185BD ld      bc,BD85  fonction Basic EXP
D523 18E5   jr      D50A     EXP (fonction)

*****
D525 0182BD ld      bc,BD82  fonction Basic LOG10
D528 18E0   jr      D50A     LOG10 (fonction)

*****
D52A 017FBD ld      bc,BD7F  fonction Basic LOG
D52D 18DB   jr      D50A     LOG (fonction)

*****
D52F 0188BD ld      bc,BD88  fonction Basic SIN
D532 18D6   jr      D50A     'SIN (fonction)

*****
D534 018BBD ld      bc,BD8B  fonction Basic COS
D537 18D1   jr      D50A     COS (fonction)

*****
D539 018EBD ld      bc,BD8E  fonction Basic TAN
D53C 18CC   jr      D50A     TAN (fonction)

*****
D53E 0191BD ld      bc,BD91  fonction Basic ATN
D541 18C7   jr      D50A     ATN (fonction)

*****
D543 52 61 6E 64 6F 6D 20 6E  'Random number seed ?'
D54B 75 6D 62 65 72 20 73 65
D553 65 64 20 3F 20 00

*****
D559 2806   jr      z,D561  instruction Basic RANDOMIZE
D55B CDFBCE call   CEFB     aller chercher expression
D55E E5     push  hl
D55F 181B   jr      D57C

```

BASIC 1.0

```

D561 E5      push   hl
D562 2143D5  ld      hl,D543   'Random number seed ?'
D565 CD41C3  call   C341      sortir
D568 CD3BCA  call   CA3B      aller chercher ligne d'entrée
D56B D26BCB  jp     nc,CB6B   touche ESC enfoncée ?
D56E CD4EC3  call   C34E      sortir LF
D571 CDA3EC  call   ECA3      lire entrée
D574 30EC    jr     nc,D562   non valable, recommencer
D576 CD61DD  call   DD61      ignorer espace, TAB et LF
D579 B7       or     a
D57A 20E6    jr     nz,D562
D57C CDECFE  call   FEEC      CREAL
D57F CD9ABD  call   BD9A      Set Random Seed
D582 E1      pop    hl
D583 C9      ret

***** RND
D584 7E      ld     a,(hl)
D585 FE28    cp     28
D587 201C    jr     nz,D5A5
D589 CD3FDD  call   DD3F      ignorer espaces
D58C CDFBCE  call   CEFB      aller chercher expression
D58F CD37DD  call   DD37      tester si encore un caractère
D592 29      db     29
D593 E5      push   hl
D594 CDECFE  call   FEEC      CREAL
D597 CD70BD  call   BD70      SGN
D59A 2005    jr     nz,D5A1   différent zéro ?
D59C CDA0BD  call   BDA0      aller chercher dernière valeur RND
D59F E1      pop    hl
D5A0 C9      ret

D5A1 FC9ABD  call   m,BD9A    Set Random Seed
D5A4 E1      pop    hl
D5A5 E5      push   hl
D5A6 CD16FF  call   FF16      fixer type sur virgule flottante
D5A9 CD9DBD  call   BD9D      RND
D5AC E1      pop    hl
D5AD C9      ret

```

BASIC 1.0

```
***** réinitialiser pointeur de variable
D5AE CDBED5 call D5BE vider table
D5B1 2A83AE ld hl,(AE83) fin du programme
D5B4 2285AE ld (AE85),hl début des variables
D5B7 2287AE ld (AE87),hl début des tableaux
D5BA 2289AE ld (AE89),hl fin des tableaux
D5BD C9 ret
```

```
***** vider tables
D5BE 21D0AD ld hl,ADDO
D5C1 3E36 ld a,36 54 = 2*27, variables + fonctions
D5C3 CDCBD5 call D5CB supprimer de ADDO à AE05
D5C6 2106AE ld hl,AE06
D5C9 3E06 ld a,06 supprimer de AE06 à AE0B
D5CB 3600 ld (hl),00 tableaux
D5CD 23 inc hl
D5CE 3D dec a
D5CF 20FA jr nz,D5CB
D5D1 C9 ret
```

```
***** supprimer flag pour FN
D5D2 210000 ld hl,0000
D5D5 2204AE ld (AE04),hl
D5D8 C9 ret
```

```
***** calculer adresse de table
D5D9 3E5B ld a,5B 'Z'+1, FN
D5DB 2A85AE ld hl,(AE85) début des variables
D5DE 2B dec hl moins 1
D5DF 44 ld b,h dans bc
D5E0 4D ld c,l
D5E1 87 add a,a fois 2
D5E2 C64E add a,4E
D5E4 6F ld l,a plus AD4E
D5E5 CEAD adc a,AD donne ADDO - AE02 pour 'A' - 'Z'
D5E7 95 sub l
D5E8 67 ld h,a
D5E9 C9 ret
```

```
***** calculer adresse de table pour tableaux
```


BASIC 1.0

D5EA	2A87AE	ld	hl,(AE87)	début des tableaux
D5ED	2B	dec	hl	moins 1
D5EE	44	ld	b,h	
D5EF	4D	ld	c,l	dans bc
D5F0	E603	and	03	
D5F2	3D	dec	a	
D5F3	87	add	a,a	
D5F4	C606	add	a,06	
D5F6	6F	ld	l,a	plus AE06
D5F7	CEAE	adc	a,AE	
D5F9	95	sub	l	
D5FA	67	ld	h,a	
D5FB	C9	ret		

***** toutes les variables sur le type REAL

D5FC	015A41	ld	bc,415A	'AZ'
D5FF	1E05	ld	e,05	'Real'
D601	79	ld	a,c	
D602	90	sub	b	nombre dans a
D603	383D	jr	c,D642	inférieur 1, 'Syntax error'
D605	E5	push	hl	
D606	3C	inc	a	
D607	21CBAD	ld	hl,ADCB	Base de la table = ADCB+'A'
D60A	0600	ld	b,00	
D60C	09	add	hl,bc	lettre pointeur dans table
D60D	73	ld	(hl),e	sauvegarder type
D60E	2B	dec	hl	
D60F	3D	dec	a	toutes les lettres
D610	20FB	jr	nz,D60D	
D612	E1	pop	hl	
D613	C9	ret		

***** instruction Basic DEFSTR

D614	1E03	ld	e,03	'chaîne'
D616	1806	jr	D61E	

***** instruction Basic DEFINT

D618	1E02	ld	e,02	'Integer'
D61A	1802	jr	D61E	

```

*****
D61C 1E05    ld    e,05    instruction Basic DEFREAL
D61E 7E      ld    a,(h1)   'Real'
D61F CD71FF  call  FF71     aller chercher lettre
D622 301E    jr    nc,D642  tester si lettre
D624 4F      ld    c,a      'Syntax error'
D625 47      ld    b,a      dans bc (de - à)
D626 CD3FDD  call  DD3F     ignorer espaces
D629 FE2D    cp    2D      '-'
D62B 200C    jr    nz,D639
D62D CD3FDD  call  DD3F     ignorer espaces
D630 CD71FF  call  FF71     tester si lettre
D633 300D    jr    nc,D642  'Syntax error'
D635 4F      ld    c,a      à
D636 CD3FDD  call  DD3F     ignorer espaces
D639 CD01D6  call  D601     fixer type de variable
D63C CD55DD  call  DD55     virgule suit ?
D63F 38DD    jr    c,D61E   oui, traiter variable suivante
D641 C9      ret

D642 1E02    ld    e,02     'Syntax error'
D644 1806    jr    D64C

D646 1E09    ld    e,09     'Subscript out of range'
D648 1802    jr    D64C

D64A 1E0A    ld    e,0A     'Array already dimensioned'
D64C C394CA   jp    CA94     sortir message d'erreur

*****

D64F FEF8     cp    F8
D651 CAAOF1  jp    z,F1A0   extension d'instruction

*****
D654 CD86D6  call  D686     instruction Basic LET
D657 D5      push  de       aller chercher variable
D658 CD37DD  call  DD37     de
D65B EF      db    EF       tester si encore un caractère
D65C CDFBCE  call  CEFB     '='
D65F 78      ld    a,b      aller chercher expression

```

BASIC 1.0

D660	E3	ex	(sp),h1	
D661	CD66D6	call	D666	affecter valeur à variable
D664	E1	pop	h1	
D665	C9	ret		
*****				affecter valeur à variable
D666	47	ld	b,a	comparer type de variable
D667	CD23FF	call	FF23	et type de résultat
D66A	B8	cp	b	
D66B	78	ld	a,b	
D66C	C4D7FE	call	nz,FED7	adapter type, sinon 'Type mismatch'
D66F	CD45FF	call	FF45	tester si chaîne
D672	C262FF	jp	nz,FF62	non, copier variable dans (h1)
D675	E5	push	h1	
D676	CD59FB	call	FB59	gestion de chaîne
D679	D1	pop	de	
D67A	C366FF	jp	FF66	accepteur pointeur sur chaîne
*****				instruction Basic DIM
D67D	CDB5D7	call	D7B5	dimensionnement
D680	CD55DD	call	DD55	virgule suit ?
D683	38F8	jr	c,D67D	oui, variable suivante
D685	C9	ret		
*****				chercher variable
D686	CD06D9	call	D906	lire nom de variable
D689	CDDBD7	call	D7DB	tester si variable dimensionnée
D68C	3842	jr	c,D6D0	aller chercher type de variable
D68E	1828	jr	D6B8	
*****				aller chercher adresse de variable
D690	CD06D9	call	D906	lire nom de variable
D693	CDDBD7	call	D7DB	tester si variable dimensionnée
D696	3838	jr	c,D6D0	aller chercher type de variable
D698	E5	push	h1	
D699	79	ld	a,c	première lettre
D69A	CDDBD5	call	D5DB	calculer position de table
D69D	CDDDED6	call	D6DE	
D6A0	182D	jr	D6CF	

BASIC 1.0

```
*****
D6A2 CD06D9 call D906 chercher fonction
D6A5 3821 jr c,D6C8 lire nom de variable
D6A7 E5 push hl
D6A8 CDD9D5 call D5D9 calculer position de table pour FN
D6AB CDD6D6 call D6DE
D6AE D43DD7 call nc,D73D mettre fonction en place
D6B1 181C jr D6CF
```

```
*****
D6B3 CD06D9 call D906 lire nom de variable
D6B6 3810 jr c,D6C8
D6B8 E5 push hl
D6B9 79 ld a,c première lettre
D6BA CDD6D5 call D5DB calculer position de table
D6BD CDD6D6 call D6DE
D6C0 3AC1B0 ld a,(B0C1) type de variable
D6C3 D449D7 call nc,D749
D6C6 1807 jr D6CF
```

```
D6C8 E5 push hl
D6C9 2A85AE ld hl,(AE85) début des variables
D6CC 2B dec hl
D6CD 19 add hl,de
D6CE EB ex de,hl
D6CF E1 pop hl
D6D0 3AC1B0 ld a,(B0C1) type de variable
D6D3 47 ld b,a
D6D4 4F ld c,a
D6D5 C9 ret
```

```
*****
D6D6 CD06D9 call D906 lire nom de variable
D6D9 CDC1E8 call E8C1 tester si variable indicée
D6DC 18F2 jr D6D0 aller chercher type de variable
```

```
*****
D6DE D5 push de
D6DF EB ex de,hl
D6E0 2A2BAE ld hl,(AE2B)
```

BASIC 1.0

```

D6E3 7C      ld      a,h
D6E4 B5      or      l
D6E5 280E    Jr      z,D6F5
D6E7 D5      push   de
D6E8 23      inc    hl
D6E9 23      inc    hl
D6EA C5      push   bc
D6EB 010000  ld     bc,0000
D6EE CD08D7  call  D708      chercher tableau
D6F1 C1      pop    bc
D6F2 3810    Jr     c,D704    trouvé ?
D6F4 D1      pop    de
D6F5 EB      ex     de,hl
D6F6 E5      push   hl
D6F7 CD08D7  call  D708      chercher tableau
D6FA 3803    Jr     c,D6FF    trouvé ?
D6FC E1      pop    hl
D6FD D1      pop    de
D6FE C9      ret

D6FF F1      pop    af
D700 E1      pop    hl
D701 C36DD7  Jp     D76D

D704 F1      pop    af
D705 F1      pop    af
D706 37      scf
D707 C9      ret

```

***** chercher tableau

```

D708 7E      ld     a,(hl)
D709 23      inc    hl
D70A 66      ld     h,(hl)
D70B 6F      ld     l,a
D70C B4      or     h
D70D C8      ret    z
D70E 09      add   hl,bc
D70F E5      push  hl
D710 23      inc   hl
D711 23      inc   hl

```

BASIC 1.0

```

D712 EB      ex      de,h1
D713 2A27AE  ld      hl,(AE27)
D716 1A      ld      a,(de)
D717 BE      cp      (hl)
D718 2014    jr      nz,D72E
D71A 23      inc     hl
D71B 13      inc     de
D71C 17      rla
D71D 30F7    jr      nc,D716
D71F EB      ex      de,h1
D720 3AC1B0  ld      a,(BOC1)    type de variable
D723 3D      dec     a
D724 AE      xor     (hl)
D725 E607    and    07
D727 2005    jr      nz,D72E
D729 EB      ex      de,h1
D72A 13      inc     de
D72B E1      pop    hl
D72C 37      scf
D72D C9      ret

D72E E1      pop    hl
D72F 18D7    jr      D708        chercher tableau

```

```

D731 F5      push   af
D732 54      ld      d,h
D733 5D      ld      e,l
D734 23      inc     hl
D735 23      inc     hl
D736 7E      ld      a,(hl)
D737 23      inc     hl
D738 17      rla
D739 30FB    jr      nc,D736
D73B F1      pop    af
D73C C9      ret

```

```

D73D 3E02    ld      a,02
D73F CD49D7 call   D749

```

BASIC 1.0

```

D742 1B      dec    de
D743 1A      ld     a,(de)
D744 F640    or     40      mettre bit 6, 'FN'
D746 12      ld     (de),a
D747 13      inc    de
D748 C9      ret

```

```

D749 D5      push   de
D74A E5      push   hl
D74B C5      push   bc
D74C F5      push   af
D74D CD77D7  call   D777
D750 F5      push   af
D751 2A87AE  ld     hl,(AE87)  début des tableaux
D754 EB      ex     de,hl
D755 CDF8F5  call   F5F8      réserver place dans zone des
                    variables
D758 CD3AF5  call   F53A      augmenter pointeur pour zone
D75B F1      pop    af
D75C CD8AD7  call   D78A
D75F F1      pop    af
D760 2B      dec    hl
D761 3600    ld     (hl),00
D763 3D      dec    a
D764 20FA    jr     nz,D760
D766 C1      pop    bc
D767 E3      ex     (sp),hl
D768 CDA5D7  call   D7A5
D76B D1      pop    de
D76C E1      pop    hl
D76D 23      inc    hl
D76E 7B      ld     a,e
D76F 91      sub    c
D770 77      ld     (hl),a
D771 23      inc    hl
D772 7A      ld     a,d
D773 98      sbc    a,b
D774 77      ld     (hl),a
D775 37      scf

```

BASIC 1.0

D776 C9 ret

D777 C603 add a,03
 D779 4F ld c,a
 D77A 2A27AE ld hl,(AE27)
 D77D 0600 ld b,00
 D77F 0C inc c
 D780 04 inc b
 D781 7E ld a,(hl)
 D782 23 inc hl
 D783 17 rla
 D784 30F9 jr nc,D77F
 D786 78 ld a,b
 D787 0600 ld b,00
 D789 C9 ret

D78A 62 ld h,d
 D78B 6B ld l,e
 D78C 09 add hl,bc
 D78D 4F ld c,a
 D78E 0600 ld b,00
 D790 E5 push hl
 D791 D5 push de
 D792 13 inc de
 D793 13 inc de
 D794 2A27AE ld hl,(AE27)
 D797 CDF2FF call FF2 ldir
 D79A 3AC1B0 ld a,(BOC1) type de variable
 D79D 3D dec a
 D79E 12 ld (de),a
 D79F 13 inc de
 D7A0 42 ld b,d
 D7A1 4B ld c,e
 D7A2 D1 pop de
 D7A3 E1 pop hl
 D7A4 C9 ret

BASIC 1.0

```

D7A5 7E      ld      a,(hl)
D7A6 12      ld      (de),a
D7A7 7B      ld      a,e
D7A8 91      sub     c
D7A9 77      ld      (hl),a
D7AA 23      inc     hl
D7AB 7E      ld      a,(hl)
D7AC F5      push    af
D7AD 7A      ld      a,d
D7AE 98      sbc     a,b
D7AF 77      ld      (hl),a
D7B0 F1      pop     af
D7B1 13      inc     de
D7B2 12      ld      (de),a
D7B3 13      inc     de
D7B4 C9      ret

```

Dimensionnement

```

D7B5 CD06D9  call    D906      aller chercher nom variable
D7B8 7E      ld      a,(hl)
D7B9 FE28    cp      28          '('
D7BB 2805    jr      z,D7C2
D7BD EE5B    xor     5B          'ET'
D7BF C242D6  jp      nz,D642     'Syntax error'
D7C2 CD5AD8  call    D85A
D7C5 E5      push    hl
D7C6 C5      push    bc
D7C7 3AC1B0  ld      a,(B0C1)    type de variable
D7CA CDEAD5  call    D5EA        calculer position table pour tableau
D7CD CD08D7  call    D708        chercher tableau
D7D0 DA4AD6  jp      c,D64A      trouvé, 'Array already dimensioned'
D7D3 C1      pop     bc
D7D4 3EFF    ld      a,FF
D7D6 CD8AD8  call    D88A
D7D9 E1      pop     hl
D7DA C9      ret

```

tester si variable dimensionnée

```

D7DB F5      push    af
D7DC 7E      ld      a,(hl)

```

BASIC 1.0

D7DD	FE28	cp	28	'('
D7DF	2810	jr	z,D7F1	
D7E1	EE5B	xor	5B	'ET'
D7E3	280C	jr	z,D7F1	
D7E5	F1	pop	af	
D7E6	D0	ret	nc	
D7E7	E5	push	hl	
D7E8	2A85AE	ld	hl,(AE85)	début des variables
D7EB	2B	dec	hl	
D7EC	19	add	hl,de	
D7ED	EB	ex	de,hl	
D7EE	E1	pop	hl	
D7EF	37	scf		
D7F0	C9	ret		
*****				variable dimensionnée
D7F1	CD5AD8	call	D85A	
D7F4	F1	pop	af	
D7F5	E5	push	hl	
D7F6	3007	jr	nc,D7FF	
D7F8	2A87AE	ld	hl,(AE87)	début des tableaux
D7FB	2B	dec	hl	
D7FC	19	add	hl,de	
D7FD	1815	jr	D814	
D7FF	C5	push	bc	
D800	D5	push	de	
D801	3AC1B0	ld	a,(B0C1)	type de variable
D804	CDEAD5	call	D5EA	calculer position table pour tableau
D807	CD08D7	call	D708	chercher tableau
D80A	300F	jr	nc,D81B	pas trouvé ?
D80C	13	inc	de	
D80D	13	inc	de	
D80E	E1	pop	hl	
D80F	CD6DD7	call	D76D	
D812	C1	pop	bc	
D813	EB	ex	de,hl	
D814	78	ld	a,b	
D815	96	sub	(hl)	
D816	C246D6	jp	nz,D646	'Subscript out of range'

BASIC 1.0

D819	180A	Jr	D825	
D81B	E1	pop	hl	
D81C	C1	pop	bc	
D81D	AF	xor	a	
D81E	CD8AD8	call	D88A	
D821	CD6DD7	call	D76D	
D824	EB	ex	de,hl	
D825	110000	ld	de,0000	
D828	46	ld	b,(hl)	nombre de dimensions
D829	23	inc	hl	
D82A	E5	push	hl	
D82B	D5	push	de	
D82C	5E	ld	e,(hl)	
D82D	23	inc	hl	limite tableau dans de
D82E	56	ld	d,(hl)	
D82F	3E02	ld	a,02	
D831	CDA0F5	call	F5A0	libérer place dans pile Basic
D834	7E	ld	a,(hl)	
D835	23	inc	hl	aller chercher index dans hl
D836	66	ld	h,(hl)	
D837	6F	ld	l,a	
D838	CDB8FF	call	FFB8	comparer hl <> de
D83B	D246D6	jp	nc,D646	'Subscript out of range'
D83E	E3	ex	(sp),hl	
D83F	CDBEBD	call	BDBE	multiplication entiers sans signe
D842	D1	pop	de	
D843	19	add	hl,de	
D844	EB	ex	de,hl	
D845	E1	pop	hl	
D846	23	inc	hl	
D847	23	inc	hl	
D848	05	dec	b	prochain index
D849	20DF	Jr	nz,D82A	
D84B	E5	push	hl	
D84C	2AC1B0	ld	hl,(BOC1)	type de variable
D84F	2600	ld	h,00	
D851	CDBEBD	call	BDBE	multiplication entiers sans signe
D854	D1	pop	de	
D855	19	add	hl,de	

BASIC 1.0

```
D856 EB      ex      de,h1
D857 E1      pop     h1
D858 37      scf
D859 C9      ret
```

```
***** lire indices
D85A D5      push   de
D85B CD3FDD  call   DD3F      ignorer espaces
D85E 3AC1B0  ld     a,(BOC1)  type de variable
D861 F5      push   af
D862 0600    ld     b,00
D864 CD7CCE  call   CE7C      aller chercher valeur 16 bits 0 -
                    32767 , index

D867 E5      push   h1
D868 3E02    ld     a,02
D86A CDB0F5  call   F5B0      réserver place dans pile Basic
D86D 73      ld     (h1),e
D86E 23      inc    h1        index sur pile Basic
D86F 72      ld     (h1),d
D870 E1      pop    h1
D871 04      inc    b
D872 CD55DD  call   DD55      virgule suit ?
D875 38ED    jr     c,D864    oui, index suivant
D877 7E      ld     a,(h1)
D878 FE29    cp     29        ')'
D87A 2805    jr     z,D881
D87C FE5D    cp     5D        'EU'
D87E C242D6  jp     nz,D642   'Syntax error'
D881 CD3FDD  call   DD3F      ignorer espaces
D884 F1      pop    af
D885 32C1B0  ld     (BOC1),a  type de variable
D888 D1      pop    de
D889 C9      ret
```

```
*****
D88A E5      push   h1
D88B 3226AE  ld     (AE26),a
D88E C5      push   bc
D88F 78      ld     a,b
D890 87      add    a,a
```

BASIC 1.0

D891	C603	add	a,03	
D893	CD77D7	call	D777	
D896	F5	push	af	
D897	2A89AE	ld	hl,(AE89)	fin des tableaux
D89A	EB	ex	de,hl	
D89B	CDF8F5	call	F5F8	r�server place dans zone variables
D89E	F1	pop	af	
D89F	CD8AD7	call	D78A	
D8A2	60	ld	h,b	
D8A3	69	ld	l,c	
D8A4	C1	pop	bc	
D8A5	D5	push	de	
D8A6	23	inc	hl	
D8A7	23	inc	hl	
D8A8	3AC1B0	ld	a,(BOC1)	type de variable
D8AB	5F	ld	e,a	
D8AC	1600	ld	d,00	
D8AE	70	ld	(hl),b	
D8AF	E5	push	hl	
D8B0	23	inc	hl	
D8B1	D5	push	de	
D8B2	3A26AE	ld	a,(AE26)	
D8B5	B7	or	a	
D8B6	110B00	ld	de,000B	11, valeur d�faut pour index
D8B9	280B	jr	z,D8C6	
D8BB	E5	push	hl	
D8BC	3E02	ld	a,02	
D8BE	CDA0F5	call	F5A0	lib�rer place dans pile Basic
D8C1	5E	ld	e,(hl)	
D8C2	23	inc	hl	
D8C3	56	ld	d,(hl)	
D8C4	13	inc	de	
D8C5	E1	pop	hl	
D8C6	73	ld	(hl),e	
D8C7	23	inc	hl	
D8C8	72	ld	(hl),d	
D8C9	23	inc	hl	
D8CA	E3	ex	(sp),hl	
D8CB	CDBEBD	call	BDBE	multiplication entiers sans signe
D8CE	DA46D6	jp	c,D646	'Subscript out of range'

BASIC 1.0

D8D1	EB	ex	de,hl	
D8D2	E1	pop	hl	
D8D3	10DC	djnz	D8B1	prochain index
D8D5	42	ld	b,d	
D8D6	4B	ld	c,e	
D8D7	54	ld	d,h	
D8D8	5D	ld	e,l	
D8D9	CDFBF5	call	F5FB	réserver place en mémoire
D8DC	2289AE	ld	(AE89),hl	fin des tableaux
D8DF	C5	push	bc	
D8E0	2B	dec	hl	
D8E1	3600	ld	(hl),00	
D8E3	0B	dec	bc	
D8E4	78	ld	a,b	
D8E5	B1	or	c	
D8E6	20F8	jr	nz,D8E0	
D8E8	C1	pop	bc	
D8E9	E1	pop	hl	
D8EA	5E	ld	e,(hl)	
D8EB	1600	ld	d,00	
D8ED	EB	ex	de,hl	
D8EE	29	add	hl,hl	
D8EF	23	inc	hl	
D8F0	09	add	hl,bc	
D8F1	EB	ex	de,hl	
D8F2	2B	dec	hl	
D8F3	2B	dec	hl	
D8F4	73	ld	(hl),e	
D8F5	23	inc	hl	
D8F6	72	ld	(hl),d	
D8F7	23	inc	hl	
D8F8	E3	ex	(sp),hl	
D8F9	EB	ex	de,hl	
D8FA	3AC1B0	ld	a.(B0C1)	type de variable
D8FD	CDEAD5	call	D5EA	calculer position table pour tableau
D900	CDA5D7	call	D7A5	
D903	D1	pop	de	
D904	E1	pop	hl	
D905	C9	ret		

BASIC 1.0

```

*****
D906 CD7FD9 call D97F aller chercher nom de variable
D909 23 inc hl déterminer type de variable
D90A 5E ld e,(hl)
D90B 23 inc hl
D90C 56 ld d,(hl)
D90D 7A ld a,d
D90E B3 or e variable déjà initialisée ?
D90F 280A jr z,D91B non
D911 23 inc hl
D912 7E ld a,(hl) ignorer lettres du nom
D913 17 rla tester bit 7
D914 30FB jr nc,D911 dernière lettre ?
D916 CD3FDD call DD3F ignorer espaces
D919 37 scf
D91A C9 ret

*****
D91B 2B dec hl
D91C 2B dec hl fixer pointeur sur type de variable
D91D EB ex de,hl
D91E C1 pop bc
D91F 2A27AE ld hl,(AE27)
D922 E5 push hl
D923 212BD9 ld hl,D92B
D926 E5 push hl
D927 C5 push bc
D928 EB ex de,hl
D929 180E jr D939

D92B E5 push hl
D92C 2A27AE ld hl,(AE27)
D92F CDACF5 call F5AC fixer pointeur de pile Basic
D932 E1 pop hl
D933 E3 ex (sp),hl
D934 2227AE ld (AE27),hl
D937 E1 pop hl
D938 C9 ret

D939 E5 push hl

```

BASIC 1.0

D93A	7E	ld	a,(hl)	
D93B	23	inc	hl	
D93C	23	inc	hl	
D93D	23	inc	hl	
D93E	4E	ld	c,(hl)	
D93F	CBA9	res	5,c	
D941	FE0B	cp	0B	
D943	3819	Jr	c,D95E	
D945	79	ld	a,c	
D946	E61F	and	1F	
D948	C60B	add	a,0B	
D94A	5F	ld	e,a	plus AE0B
D94B	CEAE	adc	a,AE	
D94D	93	sub	e	
D94E	57	ld	d,a	
D94F	1A	ld	a,(de)	
D950	32C1B0	ld	(BOC1),a	type de variable
D953	E3	ex	(sp),hl	
D954	360D	ld	(hl),0D	
D956	FE05	cp	05	
D958	2803	Jr	z,D95D	
D95A	C609	add	a,09	05 + 09 => 0D
D95C	77	ld	(hl),a	
D95D	E3	ex	(sp),hl	
D95E	EB	ex	de,hl	
D95F	3E28	ld	a,28	40
D961	CDB0F5	call	F5B0	réserver place dans pile Basic
D964	2227AE	ld	(AE27),hl	
D967	0629	ld	b,29	41
D969	05	dec	b	déjà 40 caractères ?
D96A	CA42D6	Jp	z,D642	oui, 'Syntax error'
D96D	1A	ld	a,(de)	aller chercher prochain caractère du nom
D96E	13	inc	de	
D96F	E6DF	and	DF	convertir minuscules en majuscules
D971	77	ld	(hl),a	sauvegarder dans pile Basic
D972	23	inc	hl	
D973	17	rla		dernier caractère ?
D974	30F3	Jr	nc,D969	non
D976	CDACF5	call	F5AC	fixer pointeur de pile Basic

BASIC 1.0

```
D979 EB      ex      de,h1
D97A 2B      dec     h1
D97B D1      pop     de
D97C C33FDD  jp      DD3F      ignorer espaces
```

***** déterminer type de variable

```
D97F 7E      ld      a,(h1)
D980 FE0B   cp      0B
D982 3802   jr      c,D986     inférieur 0B ?
D984 C6F7   add     a,F7        -9, 0D => 05
D986 FE04   cp      04        '!', Real-Variable ?
D988 2809   jr      z,D993     fixer type sur 'real'
D98A 3004   jr      nc,D990    'Syntax error'
D98C FE02   cp      02        '%', Integer-Variable ?
D98E 3005   jr      nc,D995    ou '$', chaîne ?
D990 C342D6  jp      D642      'Syntax error'

D993 3E05   ld      a,05       'Real'
D995 32C1B0 ld     (BOC1),a    ranger type de variable
D998 C9      ret
```

***** actualiser table des tableaux

```
D999 CDC6D5  call   D5C6        vider table pour tableaux
D99C 2A89AE  ld     hl,(AE89)   fin des tableaux
D99F EB      ex      de,h1
D9A0 2A87AE  ld     hl,(AE87)   début des tableaux
D9A3 CDB8FF  call   FFB8        comparer hl <> de
D9A6 C8      ret     z
D9A7 D5      push  de
D9A8 CD31D7  call   D731
D9AB 7E      ld     a,(h1)
D9AC 23      inc   hl
D9AD E607   and   07
D9AF 3C      inc   a
D9B0 E5      push  hl
D9B1 CDEAD5  call   D5EA        calculer position table pour tableau
D9B4 CDA5D7  call   D7A5
D9B7 E1      pop   hl
D9B8 5E      ld     e,(h1)
D9B9 23      inc   hl
```

BASIC 1.0

```
D9BA 56      ld      d,(hl)
D9BB 23      inc     hl
D9BC 19      add     hl,de
D9BD D1      pop     de
D9BE 18E3    jr      D9A3
```

***** instruction Basic ERASE

```
D9C0 CD89E9  call    E989
D9C3 CDCCD9  call    D9CC      supprimer tableau
D9C6 CD55DD  call    DD55      virgule suit ?
D9C9 38F8    jr      c,D9C3    oui, tableau suivant
D9CB C9      ret
```

***** supprimer tableau

```
D9CC CD06D9  call    D906      aller chercher nom variable
D9CF E5      push   hl
D9D0 3AC1B0  ld      a,(BOC1)  type de variable
D9D3 CDEAD5  call    D5EA      calculer position table pour tableau
D9D6 CD08D7  call    D708      chercher tableau
D9D9 E5      push   hl
D9DA EB      ex     de,hl
D9DB 1E05    ld      e,05      'Improper argument'
D9DD D294CA   jp      nc,CA94   pas initialisé, sortir message
                        d'erreur
```

```
D9E0 5E      ld      e,(hl)
D9E1 23      inc     hl        adresse du tableau dans de
D9E2 56      ld      d,(hl)
D9E3 23      inc     hl
D9E4 19      add     hl,de
D9E5 EB      ex     de,hl
D9E6 2A89AE  ld      hl,(AE89) fin des tableaux
D9E9 CDCFFF  call    FFCF      hl := hl - de
D9EC E3      ex     (sp),hl
D9ED C1      pop    bc
D9EE EB      ex     de,hl
D9EF 78      ld     a,b
D9F0 B1      or     c
D9F1 C4F2FF  call    nz,FFF2   ldir
D9F4 EB      ex     de,hl
```

BASIC 1.0

```
D9F5 2289AE ld (AE89),h1 fin des tableaux
D9F8 CD99D9 call D999 actualiser table des tableaux
D9FB E1 pop hl
D9FC C9 ret
```

```
D9FD 210000 ld hl,0000
DA00 222BAE ld (AE2B),h1
DA03 2229AE ld (AE29),h1
DA06 C9 ret
```

```
DA07 E5 push hl
DA08 2A2BAE ld hl,(AE2B)
DA0B E5 push hl
DA0C 2A29AE ld hl,(AE29)
DA0F EB ex de,hl
DA10 3E06 ld a,06
DA12 CDB0F5 call F5B0 réserver place dans pile Basic
DA15 2229AE ld (AE29),h1
DA18 73 ld (hl),e
DA19 23 inc hl
DA1A 72 ld (hl),d
DA1B 23 inc hl
DA1C AF xor a
DA1D 77 ld (hl),a
DA1E 23 inc hl
DA1F 77 ld (hl),a
DA20 23 inc hl
DA21 D1 pop de
DA22 73 ld (hl),e
DA23 23 inc hl
DA24 72 ld (hl),d
DA25 E1 pop hl
DA26 C9 ret
```

```
DA27 E5 push hl
DA28 2A29AE ld hl,(AE29)
DA2B 222BAE ld (AE2B),h1
```

BASIC 1.0

```

DA2E E1      pop    hl
DA2F C9      ret

*****
DA30 E5      push   hl
DA31 2A29AE ld     hl,(AE29)
DA34 CDACF5 call   F5AC      fixer pointeur de pile Basic
DA37 5E      ld     e,(hl)
DA38 23      inc    hl
DA39 56      ld     d,(hl)
DA3A 23      inc    hl
DA3B EB      ex     de,hl
DA3C 2229AE ld     (AE29),hl
DA3F EB      ex     de,hl
DA40 23      inc    hl
DA41 23      inc    hl
DA42 5E      ld     e,(hl)
DA43 23      inc    hl
DA44 56      ld     d,(hl)
DA45 EB      ex     de,hl
DA46 222BAE ld     (AE2B),hl
DA49 E1      pop    hl
DA4A C9      ret

*****
DA4B E5      push   hl
DA4C 3E02    ld     a,02
DA4E CDB0F5 call   F5B0      réserver place dans pile Basic
DA51 E3      ex     (sp),hl
DA52 CD7FD9 call   D97F      déterminer type de variable
DA55 CD39D9 call   D939
DA58 E3      ex     (sp),hl
DA59 EB      ex     de,hl
DA5A 2A29AE ld     hl,(AE29)
DA5D 23      inc    hl
DA5E 23      inc    hl
DA5F 010000 ld     bc,0000
DA62 CDA5D7 call   D7A5
DA65 3AC1B0 ld     a,(BOC1)   type de variable
DA68 47      ld     b,a

```

BASIC 1.0

```

,9 3C      inc    a
5A CDB0F5  call   F5B0      réserver place dans pile Basic
6D 78      ld     a,b
46E 3D     dec    a
A6F 77     ld     (hl),a
A70 23     inc    hl
JA71 EB    ex     de,hl
JA72 E1    pop    hl
DA73 C9    ret

```

```

DA74 2A29AE ld     hl,(AE29)
DA77 7C      ld     a,h
DA78 B5      or     l
DA79 280E    jr     z,DA89
DA7B 4E      ld     c,(hl)
DA7C 23     inc    hl
DA7D 46     ld     b,(hl)
DA7E 23     inc    hl
DA7F C5     push   bc
DA80 010000 ld     bc,0000
DA83 CDCEDA  call   DACE
DA86 E1     pop    hl
D A 8 7      1 8 E E

```

J r

D A 7 7

```

DA89 01411A ld     bc,1A41      26 lettres, 'A'
DA8C C5     push   bc
DA8D 79     ld     a,c      première lettre du nom
DA8E CDDBD5 call   D5DB      calculer position de table
DA91 CDCEDA call   DACE
DA94 C1     pop    bc
DA95 0C     inc    c      lettre suivante
DA96 05     dec    b      déjà toutes les lettres ?
DA97 20F3   jr     nz,DA8C
DA99 3E03   ld     a,03
DA9B CDEAD5 call   D5EA      calculer position table pour tableau
DA9E 4E     ld     c,(hl)
DA9F 23     inc    hl
DAA0 46     ld     b,(hl)
DAA1 78     ld     a,b

```

BASIC 1.0

DAA2	B1	or	c	
DAA3	C8	ret	z	
DAA4	2A87AE	ld	hl,(AE87)	début des tableaux
DAA7	2B	dec	hl	
DAA8	09	add	hl,bc	
DAA9	E5	push	hl	
DAAA	D5	push	de	
DAAB	CD31D7	call	D731	
DAAE	D1	pop	de	
DAAF	23	inc	hl	
DAB0	4E	ld	c,(hl)	
DAB1	23	inc	hl	
DAB2	46	ld	b,(hl)	
DAB3	23	inc	hl	
DAB4	E5	push	hl	
DAB5	09	add	hl,bc	
DAB6	E3	ex	(sp),hl	
DAB7	4E	ld	c,(hl)	
DAB8	23	inc	hl	
DAB9	0600	ld	b,00	
DABB	09	add	hl,bc	
DABC	09	add	hl,bc	
DABD	C1	pop	bc	
DABE	CDBEFF	call	FFBE	comparer hl <> bc
DAC1	2808	jr	z,DACB	
DAC3	CDE7DA	call	DAE7	
DAC6	23	inc	hl	
DAC7	23	inc	hl	
DAC8	23	inc	hl	
DAC9	18F3	jr	DABE	
DACB	E1	pop	hl	
DACC	18D0	jr	DA9E	
DACE	7E	ld	a,(hl)	
DACF	23	inc	hl	
DAD0	66	ld	h,(hl)	
DAD1	6F	ld	l,a	
DAD2	B4	or	h	
DAD3	C8	ret	z	

BASIC 1.0

```

DAD4 09      add    hl,bc
DAD5 E5      push   hl
DAD6 D5      push   de
DAD7 CD31D7  call   D731
DADA D1      pop    de
DADB 7E      ld     a,(hl)
DADC 23      inc    hl
DADD E607    and    07
DADF FE02    cp     02
DAE1 CCE7DA  call   z,DAE7
DAE4 E1      pop    hl
DAE5 18E7    jr     DACE

DAE7 C5      push   bc
DAE8 D5      push   de
DAE9 E5      push   hl
DAEA 7E      ld     a,(hl)
DAEB 23      inc    hl
DAEC 4E      ld     c,(hl)
DAED 23      inc    hl
DAEE 46      ld     b,(hl)
DAEF EB      ex     de,hl
DAF0 B7      or     a
DAF1 C4F8FF  call   nz,FFF8      jp (hl)
DAF4 E1      pop    hl
DAF5 D1      pop    de
DAF6 C1      pop    bc
DAF7 C9      ret

***** instruction Basic LINE
DAF8 CD37DD  call   DD37      tester si encore un caractère
DAFB A3      db     A3      'INPUT'
DAFC CDCBC1  call   C1CB      aller chercher numéro de canal
DAFF F5      push   af
DB00 CD89DB  call   DB89      sortir éventuelle chaîne dialogue
DB03 CD86D6  call   D686      chercher variable
DB06 CD3CFF  call   FF3C      type 'chaîne', sinon 'Type mismatch'
DB09 E5      push   hl
DB0A D5      push   de
DB0B CD1ADB  call   DB1A      aller chercher entrée dans appareil

```

BASIC 1.0

				actif
DB0E	CDDCF7	call	F7DC	entrer chaîne dans pile du descripteur
DB11	E1	pop	hl	
DB12	CD6FD6	call	D66F	affecter résultat à variable
DB15	E1	pop	hl	
DB16	F1	pop	af	
DB17	C3AFC1	jp	C1AF	réinitialiser numéro canal
*****				aller chercher entrée dans appareil actif
DB1A	CDCOC1	call	C1C0	
DB1D	D266DC	jp	nc,DC66	aller chercher entrée de cassette
DB20	CDA2C1	call	C1A2	
DB23	F5	push	af	
DB24	CDADDB	call	DBAD	aller chercher entrée du clavier
DB27	F1	pop	af	
DB28	C3A2C1	jp	C1A2	
*****				instruction Basic INPUT
DB2B	CDCBC1	call	C1CB	aller chercher numéro de canal
DB2E	F5	push	af	
DB2F	CD47DB	call	DB47	aller chercher entrée et convertir
DB32	D5	push	de	
DB33	CD86D6	call	D686	aller chercher variable
DB36	E3	ex	(sp),hl	
DB37	3E00	ld	a,00	
DB39	CDBCDB	call	DBBC	
DB3C	E3	ex	(sp),hl	
DB3D	CD55DD	call	DD55	tester si virgule
DB40	38F1	jr	c,DB33	
DB42	D1	pop	de	
DB43	F1	pop	af	
DB44	C3AFC1	jp	C1AF	
*****				aller chercher entrée et convertir
DB47	CDCOC1	call	C1C0	
DB4A	303D	jr	nc,DB89	sortir éventuelle chaîne dialogue
DB4C	CDA2C1	call	C1A2	
DB4F	F5	push	af	
DB50	E5	push	hl	

BASIC 1.0

DB51	CD89DB	call	DB89	sortir éventuelle chaîne dialogue
DB54	3E3F	ld	a,3F	'?'
DB56	D456C3	call	nc,C356	
DB59	3E20	ld	a,20	'5'
DB5B	D456C3	call	nc,C356	
DB5E	E5	push	hl	
DB5F	CDADDB	call	DBAD	aller chercher entrée du clavier
DB62	EB	ex	de,hl	
DB63	E1	pop	hl	
DB64	CDD3DB	call	DBD3	
DB67	3809	jr	c,DB72	
DB69	2177DB	ld	hl,DB77	'?Redo from start'
DB6C	CD41C3	call	C341	sortir
DB6F	E1	pop	hl	
DB70	18DE	jr	DB50	
DB72	F1	pop	af	
DB73	F1	pop	af	
DB74	C3A2C1	jp	C1A2	

DB77	3F 52 65 64 6F 20 66 72	'?Redo from start'
DB7F	6F 6D 20 73 74 61 72 74	
DB87	0A 00	

DB89	7E	ld	a,(hl)	sortir éventuelle chaîne dialogue
DB8A	FE3B	cp	3B	','
DB8C	322DAE	ld	(AE2D),a	ranger signe séparation
DB8F	CC3FDD	call	z,DD3F	ignorer espaces
DB92	EE22	xor	22	'''
DB94	C0	ret	nz	
DB95	CDCBF7	call	F7CB	lire chaîne dialogue
DB98	CDC0C1	call	C1C0	
DB9B	F5	push	af	
DB9C	DC28F8	call	c,F828	sortir chaîne
DB9F	F1	pop	af	
DBA0	D4DAFB	call	nc,FBDA	aller chercher paramètres chaîne
DBA3	CD55DD	call	DD55	virgule suit ?
DBA6	D8	ret	c	oui

BASIC 1.0

```
DBA7 CD37DD call DD37 tester si encore un caractère
DBAA 3B db 3B ';'
DBAB B7 or a
DBAC C9 ret
```

```
***** aller chercher entrée du clavier
DBAD CD3BCA call CA3B aller chercher ligne d'entrée
DBB0 D26BCB jp nc,CB6B ESC enfoncée ?
DBB3 3A2DAE ld a,(AE2D) signe de séparation
DBB6 FE3B cp 3B ';'
DBB8 C44EC3 call nz,C34E pas ';', nouvelle ligne (sortir LF)
DBBB C9 ret
```

```
*****
DBBC D5 push de
DBBD CD02DC call DC02
DBC0 300C jr nc,DBCE
DBC2 E3 ex (sp),hl
DBC3 CD66D6 call D666 affecter valeur à variable
DBC6 E1 pop hl
DBC7 7E ld a,(hl)
DBC8 23 inc hl
DBC9 B7 or a
DBCA C8 ret z
DBCB EE2C xor 2C ','
DBCD C9 ret

DBCE 1E0D ld e,0D 'Type mismatch'
DBD0 C394CA jp CA94 sortir message d'erreur
```

```
*****
DBD3 D5 push de
DBD4 E5 push hl
DBD5 D5 push de
DBD6 CDD6D6 call D6D6 aller chercher nom et type de
variable
DBD9 E3 ex (sp),hl
DBDA AF xor a
DBDB CD02DC call DC02
DBDE 301E jr nc,DBFE
```

BASIC 1.0

DBE0	FE03	cp	03	'chaîne'
DBE2	CCDAFB	call	z,FBDA	oui, aller chercher paramètre de chaîne
DBE5	E3	ex	(sp),h1	
DBE6	CD55DD	call	DD55	virgule suit ?
DBE9	E3	ex	(sp),h1	
DBEA	300B	Jr	nc,DBF7	non
DBEC	CD61DD	call	DD61	ignorer espace, TAB et LF
DBEF	EE2C	xor	2C	','
DBF1	200B	Jr	nz,DBFE	
DBF3	23	inc	h1	
DBF4	E3	ex	(sp),h1	
DBF5	18DF	Jr	DBD6	
DBF7	CD61DD	call	DD61	ignorer espace, TAB et LF
DBFA	B7	or	a	
DBFB	2001	Jr	nz,DBFE	
DBFD	37	scf		
DBFE	E1	pop	h1	
DBFF	E1	pop	h1	
DC00	D1	pop	de	
DC01	C9	ret		
DC02	5F	ld	e,a	
DC03	CD45FF	call	FF45	tester si chaîne
DC06	57	ld	d,a	
DC07	D5	push	de	
DC08	2006	Jr	nz,DC10	
DC0A	CD21DC	call	DC21	
DC0D	37	scf		
DC0E	1809	Jr	DC19	
DC10	CDC0C1	call	C1C0	
DC13	D438DC	call	nc,DC38	
DC16	CDA3EC	call	ECA3	
DC19	F5	push	af	
DC1A	DC61DD	call	c,DD61	ignorer espace, TAB et LF
DC1D	F1	pop	af	
DC1E	D1	pop	de	
DC1F	7A	ld	a,d	

BASIC 1.0

DC20	C9	ret		
DC21	CDC0C1	call	C1C0	
DC24	3806	Jr	c,DC2C	
DC26	CD47DC	call	DC47	
DC29	C3DCF7	Jp	F7DC	entrer chaîne dans descripteur de chaîne
DC2C	CD61DD	call	DD61	ignorer espace, TAB et LF
DC2F	FE22	cp	22	'''
DC31	CACBF7	Jp	z,F7CB	lire chaîne
DC34	7B	ld	a,e	
DC35	C3E6F7	Jp	F7E6	
DC38	CD9DDC	call	DC9D	
DC3B	3005	Jr	nc,DC42	'EOF met'
DC3D	11C6DC	ld	de,DCC6	
DC40	182C	Jr	DC6E	
DC42	1E18	ld	e,18	'EOF met'
DC44	C394CA	Jp	CA94	sortir message d'erreur
DC47	CD9DDC	call	DC9D	
DC4A	30F6	Jr	nc,DC42	'EOF met'
DC4C	FE22	cp	22	'''
DC4E	2805	Jr	z,DC55	
DC50	11CADC	ld	de,DCCA	
DC53	1819	Jr	DC6E	
DC55	CDA8DC	call	DCA8	
DC58	1163DC	ld	de,DC63	
DC5B	3811	Jr	c,DC6E	
DC5D	21A4AC	ld	h1,ACA4	début du buffer d'entrée
DC60	3600	ld	(h1),00	premier caractère égale 00
DC62	C9	ret		
DC63	FE22	cp	22	'''
DC65	C9	ret		
DC66	CDA8DC	call	DCA8	

BASIC 1.0

DC69	30D7	jr	nc,DC42	'EOF met'
DC6B	11CDDC	ld	de,DCCD	
DC6E	21A4AC	ld	hl,ACA4	début du buffer d'entrée
DC71	E5	push	hl	
DC72	06FF	ld	b,FF	
DC74	CDFBFF	call	FFFB	jp (de)
DC77	280C	jr	z,DC85	
DC79	77	ld	(hl),a	
DC7A	23	inc	hl	
DC7B	05	dec	b	
DC7C	2805	jr	z,DC83	
DC7E	CDA8DC	call	DCA8	
DC81	38F1	jr	c,DC74	
DC83	F6FF	or	FF	
DC85	3600	ld	(hl),00	
DC87	E1	pop	hl	
DC88	C0	ret	nz	
DC89	FE0D	cp	0D	CR
DC8B	C8	ret	z	
DC8C	FE22	cp	22	'''
DC8E	C4D0DC	call	nz,DCD0	
DC91	C0	ret	nz	
DC92	CD9DDC	call	DC9D	
DC95	D0	ret	nc	
DC96	CDCADC	call	DCCA	
DC99	C414C4	call	nz,C414	CAS RETURN
DC9C	C9	ret		
DC9D	CDA8DC	call	DCA8	
DCA0	D0	ret	nc	
DCA1	CDD0DC	call	DCD0	
DCA4	28F7	jr	z,DC9D	
DCA6	37	scf		
DCA7	C9	ret		
DCA8	CD24C4	call	C424	lire et entrer un caractère
DCAB	D0	ret	nc	
DCAC	C5	push	bc	
DCAD	FE0D	cp	0D	CR
DCAF	060A	ld	b,0A	LF

BASIC 1.0

DCB1	2805	jr	z,DCB8	
DCB3	B8	cp	b	
DCB4	200D	jr	nz,DCC3	
DCB6	060D	ld	b,OD	CR
DCB8	4F	ld	c,a	
DCB9	CD24C4	call	C424	lire et entrer un caractère
DCBC	3004	jr	nc,DCC2	
DCBE	B8	cp	b	
DCBF	C414C4	call	nz,C414	CAS RETURN
DCC2	79	ld	a,c	
DCC3	C1	pop	bc	
DCC4	37	scf		
DCC5	C9	ret		
DCC6	CDB0DC	call	DCD0	
DCC9	C8	ret	z	
DCCA	FE2C	cp	2C	','
DCCC	C8	ret	z	
DCCD	FE0D	cp	0D	CR
DCCF	C9	ret		
DCD0	FE20	cp	20	'5'
DCD2	C8	ret	z	
DCD3	FE09	cp	09	TAB
DCD5	C8	ret	z	
DCD6	FE0A	cp	0A	LF
DCD8	C9	ret		
*****		instruction Basic RESTORE		
DCD9	280A	jr	z,DCE5	
DCDB	CDE1CE	call	CEE1	aller chercher numéro ligne dans de
DCDE	E5	push	h1	
DCDF	CD9AE7	call	E79A	BASIC-Zeile de suchen
DCE2	2B	dec	h1	
DCE3	182D	jr	DD12	fixer pointeur de DATA
DCE5	E5	push	h1	
DCE6	2A81AE	ld	h1,(AE81)	début de programme
DCE9	1827	jr	DD12	comme pointeur de DATA

BASIC 1.0

```

***** instruction Basic READ
DCEB E5      push   hl
DCEC 2A30AE ld     hl,(AE30)  pointeur de DATA
DCEF CD17DD  call   DD17      aller chercher prochain élément DATA
DCF2 E3      ex      (sp),hl
DCF3 CD86D6  call   D686      chercher variable
DCF6 E3      ex      (sp),hl
DCF7 23      inc     hl
DCF8 3E3A    ld     a,3A      ':'
DCFA CDBCDB  call   DBBC
DCFD 2B      dec     hl
DCFE 280B    jr     z,DD0B
DD00 2A2EAE  ld     hl,(AE2E)  adresse ligne pendant instruction
                        READ
DD03 CDCEDD  call   DDCE      fixer adresse ligne actuelle
DD06 1E02    ld     e,02      'Syntax error'
DD08 C394CA  jp     CA94      sortir message d'erreur

DD0B E3      ex      (sp),hl
DD0C CD55DD  call   DD55      virgule suit ?
DD0F E3      ex      (sp),hl
DD10 38DD    jr     c,DCEF     oui
DD12 2230AE  ld     (AE30),hl  pointeur DATA
DD15 E1      pop    hl
DD16 C9      ret

DD17 7E      ld     a,(hl)
DD18 FE2C    cp     2C        ','

```

BASIC 1.0

DD1A	C8	ret	z	
DD1B	CDEF E8	call	E8EF	ignorer reste de la ligne
DD1E	B7	or	a	fin de ligne ?
DD1F	200E	Jr	nz,DD2F	non
DD21	23	inc	hl	
DD22	7E	ld	a,(hl)	longueur de ligne
DD23	23	inc	hl	
DD24	B6	or	(hl)	zéro, fin de programme ?
DD25	23	inc	hl	
DD26	1E04	ld	e,04	'DATA exhausted'
DD28	CA94CA	jp	z,CA94	sortir message d'erreur
DD2B	222EAE	ld	(AE2E),hl	adresse de ligne pendant instruction READ
DD2E	23	inc	hl	
DD2F	CD3FDD	call	DD3F	ignorer espaces
DD32	FE8C	cp	8C	'DATA'
DD34	20E5	Jr	nz,DD1B	
DD36	C9	ret		
*****				tester si encore un caractère
DD37	E3	ex	(sp),hl	pointeur sur après instruction CALL
DD38	7E	ld	a,(hl)	aller chercher caractère suivant
DD39	23	inc	hl	
DD3A	E3	ex	(sp),hl	augmenter adresse de retour
DD3B	BE	cp	(hl)	comparer caractère avec texte programme
DD3C	C2C6DD	jp	nz,DDC6	différent, 'Syntax error'
*****				ignorer espaces
DD3F	23	inc	hl	
DD40	7E	ld	a,(hl)	
DD41	FE20	cp	20	'5'
DD43	28FA	Jr	z,DD3F	ignorer espaces
DD45	FE01	cp	01	
DD47	D0	ret	nc	
DD48	B7	or	a	fin de ligne, Z=1
DD49	C9	ret		
*****				fin de l'instruction, sinon 'Syntax error'

BASIC 1.0

```
DD4A 7E      ld      a,(h1)      caractère actuel
DD4B FE02    cp      02      inférieur 2 ?
DD4D D8      ret      c          ok
DD4E C3C6DD  ^jp     DDC6      sinon 'Syntax error'
```

```
***** tester si fin d'instruction
DD51 7E      ld      a,(h1)      caractère actuel
DD52 FE02    cp      02      inférieur 2 ?
DD54 C9      ret
```

```
***** tester si prochain caractère = virgule
DD55 2B      dec     hl
DD56 CD3FDD  call   DD3F      ignorer espaces
DD59 EE2C    xor     2C      ','
DD5B C0      ret     nz       pas trouvé, c=0
DD5C CD3FDD  call   DD3F      ignorer espaces
DD5F 37      scf
DD60 C9      ret     trouvé, c=1
```

```
***** ignorer espace, TAB et LF
DD61 7E      ld      a,(h1)
DD62 23      inc     hl
DD63 FE20    cp      20      '5'
DD65 28FA    jr     z,DD61
DD67 FE09    cp      09      TAB
DD69 28F6    jr     z,DD61
DD6B FE0A    cp      0A      LF
DD6D 28F2    jr     z,DD61
DD6F 2B      dec     hl
DD70 C9      ret
```

```
***** boucle de l'interpréteur
DD71 2A34AE  ld      hl,(AE34)  adresse de l'instruction actuelle
DD74 EB      ex      de,hl     ranger pointeur de programme
DD75 2A8BB0  ld      hl,(B08B)  pointeur de pile Basic
DD78 2232AE  ld      (AE32),hl  mémoire pour pointeur de pile Basic
DD7B EB      ex      de,hl     pointeur de programme
DD7C 2234AE  ld      (AE34),hl  comme adresse de l'instruction act.
DD7F CD21B9  call   B921      KL POLL SYNCHRONOUS
DD82 DC07C8  call   c,C807     traitement Event AFTER/EVERY
```

BASIC 1.0

DD85	CD3FDD	call	DD3F	ignorer espaces
DD88	C4ABDD	call	nz,DDAB	exécuter instruction Basic
DD8B	7E	ld	a,(hl)	lire texte programme
DD8C	FE01	cp	01	':', fin de l'instruction ?
DD8E	28E4	jr	z,DD74	oui
DD90	3034	jr	nc,DDC6	'Syntax error'
DD92	23	inc	hl	
DD93	7E	ld	a,(hl)	
DD94	23	inc	hl	longueur de ligne
DD95	B6	or	(hl)	égale zéro ?
DD96	23	inc	hl	
DD97	280F	jr	z,DDA8	oui, à l'instruction END
DD99	2236AE	ld	(AE36),hl	ranger adresse de ligne actuelle
DD9C	23	inc	hl	
DD9D	3A38AE	ld	a,(AE38)	flag TRAC mis ?
DDA0	B7	or	a	
DDA1	28D1	jr	z,DD74	non
DDA3	CDEBDD	call	DDEB	routine TRACE
DDA6	18CC	jr	DD74	à la boucle de l'interpréteur
DDA8	C376CB	jp	CB76	à l'instruction END
*****				exécuter instruction Basic
DDAB	87	add	a,a	token par 2
DDAC	D24FD6	jp	nc,D64F	tester si extension d'instruction
DDAF	FEB9	cp	B9	
DDB1	3010	jr	nc,DDC3	token non valable, 'Syntax error'
DDB3	EB	ex	de,hl	
DDB4	C601	add	a,01	
DDB6	6F	ld	l,a	plus DE01 (adresse de table)
DDB7	CEDE	adc	a,DE	
DDB9	95	sub	l	
DDBA	67	ld	h,a	
DDBB	4E	ld	c,(hl)	
DDBC	23	inc	hl	
DDBD	46	ld	b,(hl)	
DDBE	C5	push	bc	adresse de l'instruction sur pile
DDBF	EB	ex	de,hl	
DDC0	C33FDD	jp	DD3F	ignorer espaces, saut à instruction

BASIC 1.0

```

*****
DDC3 CD07AC call AC07 ret
DDC6 1E02 ld e,02 'Syntax error'
DDC8 C394CA jp CA94 sortir message d'erreur

***** adresse de ligne actuelle sur zéro
DDCB 210000 ld hl,0000
DDCE 2236AE ld (AE36),hl adresse de ligne actuelle
DDD1 C9 ret

***** charger adresse de ligne actuelle
DDD2 2A36AE ld hl,(AE36) adresse de ligne actuelle
DDD5 C9 ret

***** aller chercher test mode direct / numéro de ligne
DDD6 2A36AE ld hl,(AE36) adresse de ligne actuelle
DDD9 7C ld a,h
DDDA B5 or l
DDDB C8 ret z zéro, mode direct
DDDC 7E ld a,(hl)
DDDD 23 inc hl
DDDE 66 ld h,(hl) numéro de ligne dans hl
DDDF 6F ld l,a
DDE0 37 scf
DDE1 C9 ret

***** instruction Basic TRON
DDE2 3EFF ld a,FF
DDE4 1801 Jr DDE7

***** instruction Basic TROFF
DDE6 AF xor a
DDE7 3238AE ld (AE38),a flag TRACE
DDEA C9 ret

***** Routine TRACE
DDEB 3E5B ld a,5B 'ET'
DDED CD56C3 call C356 sortir
DDF0 E5 push hl
DDF1 2A36AE ld hl,(AE36) adresse de ligne actuelle

```

BASIC 1.0

DDF4	7E	ld	a,(hl)	
DDF5	23	inc	hl	numéro de ligne dans hl
DDF6	66	ld	h,(hl)	
DDF7	6F	ld	l,a	
DDF8	CD79EE	call	EE79	sortir numéro de ligne
DDFB	E1	pop	hl	
DDFC	3E5D	ld	a,5D	'EU'
DDFE	C356C3	jp	C356	sortir

***** adresses des instructions Basic

DE01	71C9	dw	C971	80	AFTER
DE03	DFC0	dw	C0DF	81	AUTO
DE05	21C2	dw	C221	82	BORDER
DE07	BAF1	dw	F1BA	83	CALL
DE09	46D2	dw	D246	84	CAT
DE0B	3CEA	dw	EA3C	85	CHAIN
DE0D	32C1	dw	C132	86	CLEAR
DE0F	B5C4	dw	C4B5	87	CLG
DE11	98D2	dw	D298	88	CLOSEIN
DE13	A1D2	dw	D2A1	89	CLOSEOUT
DE15	5AC2	dw	C25A	8A	CLS
DE17	C0CB	dw	CBC0	8B	CONT
DE19	EFE8	dw	E8EF	8C	DATA
DE1B	17D1	dw	D117	8D	DEF
DE1D	18D6	dw	D618	8E	DEFINT
DE1F	1CD6	dw	D61C	8F	DEFREAL
DE21	14D6	dw	D614	90	DEFSTR
DE23	E7D4	dw	D4E7	91	DEG
DE25	28E7	dw	E728	92	DELETE
DE27	7DD6	dw	D67D	93	DIM
DE29	C6C4	dw	C4C6	94	DRAW
DE2B	CBC4	dw	C4CB	95	DRAWR
DE2D	52C0	dw	C052	96	EDIT
DE2F	F3E8	dw	E8F3	97	ELSE
DE31	65CB	dw	CB65	98	END
DE33	85D3	dw	D385	99	ENT
DE35	4ED3	dw	D34E	9A	ENV
DE37	C0D9	dw	D9C0	9B	ERASE
DE39	8FCA	dw	CA8F	9C	ERROR
DE3B	79C9	dw	C979	9D	EVERY

BASIC 1.0

DE3D	29C5	dw	C529	9E	FOR
DE3F	EDC6	dw	C6ED	9F	GOSUB
DE41	E8C6	dw	C6E8	A0	GOTO
DE43	C7C6	dw	C6C7	A1	IF
DE45	2AC2	dw	C22A	A2	INK
DE47	2BDB	dw	DB2B	A3	INPUT
DE49	39D4	dw	D439	A4	KEY
DE4B	54D6	dw	D654	A5	LET
DE4D	F8DA	dw	DAF8	A6	LINE
DE4F	F7E0	dw	E0F7	A7	LIST
DE51	F6E9	dw	E9F6	A8	LOAD
DE53	D2C2	dw	C2D2	A9	LOCATE
DE55	EFF4	dw	F4EF	AA	MEMORY
DE57	A6EA	dw	EAA6	AB	MERGE
DE59	93F9	dw	F993	AC	MID\$
DE5B	4FC2	dw	C24F	AD	MODE
DE5D	05C5	dw	C505	AE	MOVE
DE5F	OAC5	dw	C50A	AF	MOVER
DE61	FBC5	dw	C5FB	B0	NEXT
DE63	2BC1	dw	C12B	B1	NEW
DE65	E3C7	dw	C7E3	B2	ON
DE67	CBC8	dw	C8CB	B3	ON BREAK
DE69	F8CB	dw	CBF8	B4	ON ERROR GOTO 0
DE6B	40C9	dw	C940	B5	ON SQ
DE6D	5FD2	dw	D25F	B6	OPENIN
DE6F	56D2	dw	D256	B7	OPENOUT
DE71	8CC4	dw	C48C	B8	ORIGIN
DE73	77F1	dw	F177	B9	OUT
DE75	OAC2	dw	C20A	BA	PAPER
DE77	12C2	dw	C212	BB	PEN
DE79	D0C4	dw	C4D0	BC	PLOT
DE7B	D5C4	dw	C4D5	BD	PLOTR
DE7D	5FF1	dw	F15F	BE	POKE
DE7F	FDF1	dw	F1FD	BF	PRINT
DE81	F3E8	dw	E8F3	C0	'
DE83	EBD4	dw	D4EB	C1	RAD
DE85	59D5	dw	D559	C2	RANDOMIZE
DE87	EBDC	dw	DCEB	C3	READ
DE89	1ED3	dw	D31E	C4	RELEASE
DE8B	F3E8	dw	E8F3	C5	REM

BASIC 1.0

DE8D	D9E7	dw	E7DF	C6	RENUM
DE8F	D9DC	dw	DCD9	C7	RESTORE
DE91	03CC	dw	CC03	C8	RESUME
DE93	0FC7	dw	C70F	C9	RETURN
DE95	BDE9	dw	E9BD	CA	RUN
DE97	09EC	dw	EC09	CB	SAVE
DE99	C0D2	dw	D2C0	CC	SOUND
DE9B	94D4	dw	D494	CD	SPEED
DE9D	5ACB	dw	CB5A	CE	STOP
DE9F	9DF6	dw	F69D	CF	SYMBOL
DEA1	19C3	dw	C319	D0	TAG
DEA3	20C3	dw	C320	D1	TAG OFF
DEA5	E6DD	dw	DDE6	D2	TRON
DEA7	E2DD	dw	DDE2	D3	TROFF
DEA9	7DF1	dw	F17D	D4	WAIT
DEAB	76C7	dw	C776	D5	WEND
DEAD	47C7	dw	C747	D6	WHILE
DEAF	E3C3	dw	C3E3	D7	WIDTH
DEB1	E1C2	dw	C2E1	D8	WINDOW
DEB3	7BF4	dw	F47B	D9	ZONE
DEB5	F6F1	dw	F1F6	DA	WRITE
DEB7	E1C8	dw	C8E1	DB	DI
DEB9	E7C8	dw	C8E7	DC	EI

DEBB	D5	push	de	
DEBC	EB	ex	de,h1	
DEBD	2A7FAE	ld	h1,(AE7F)	début de la Ram libre
DECO	EB	ex	de,h1	
DEC1	D5	push	de	
DEC2	AF	xor	a	
DEC3	3239AE	ld	(AE39),a	
DEC6	012C01	ld	bc,012C	max. 300 caractères
DEC9	CDE1DE	call	DEE1	aller chercher caractère dans buffer d'entrée
DECC	7E	ld	a,(h1)	
DECD	B7	or	a	dernier caractère ?
DECE	20F9	jr	nz,DEC9	non
DED0	3E2D	ld	a,2D	
DED2	91	sub	c	301 - état compteur

BASIC 1.0

DED3	4F	ld	c,a	
DED4	3E01	ld	a,01	égale longueur de ligne
DED6	98	sbc	a,b	
DED7	47	ld	b,a	dans b
DED8	AF	xor	a	
DED9	12	ld	(de),a	
DEDA	13	inc	de	trois fois zéro pour terminer
DEDB	12	ld	(de),a	
DEDC	13	inc	de	
DEDD	12	ld	(de),a	
DEDE	E1	pop	hl	
DEDF	D1	pop	de	
DEEO	C9	ret		

***** aller chercher caractère dans buffer d'entrée

DEE1	CD10AC	call	AC10	ret
DEE4	7E	ld	a,(hl)	
DEE5	B7	or	a	dernier caractère ?
DEE6	C8	ret	z	oui
DEE7	CD71FF	call	FF71	lettre ?
DEEA	381D	jr	c,DF09	oui
DEEC	CD7FFF	call	FF7F	numerique ?
DEEF	DAFFDF	jp	c,DFFF	oui
DEF2	FE26	cp	26	'&' ?
DEF4	CA5AEO	jp	z,E05A	oui
DEF7	23	inc	hl	
DEF8	FE80	cp	80	token ?
DEFA	D0	ret	nc	oui
DEFB	FE20	cp	20	'5'
DEFD	C280EO	jp	nz,E080	
DF00	3A00AC	ld	a,(AC00)	ignorer espaces supplémentaires ?
DF03	B7	or	a	
DF04	C0	ret	nz	oui
DF05	3E20	ld	a,20	'5'
DF07	181C	jr	DF25	écrire dans buffer
DF09	CD4EDF	call	DF4E	
DF0C	D8	ret	c	
DF0D	FEC5	cp	C5	'REM'
DF0F	CAEDEO	jp	z,E0ED	

BASIC 1.0

DF12	E5	push	hl	
DF13	2130DF	ld	hl,DF30	adresse de base de la table
DF16	CDAAFF	call	FFAA	parcourir table
DF19	E1	pop	hl	
DF1A	3819	Jr	c,DF35	trouvé ? alors ne pas convertir le reste
DF1C	F5	push	af	
DF1D	FE97	cp	97	'ELSE'
DF1F	3E01	ld	a,01	
DF21	CC25DF	call	z,DF25	écrire dans le buffer
DF24	F1	pop	af	
DF25	12	ld	(de),a	écrire un caractère dans le buffer
DF26	13	inc	de	augmenter le pointeur de buffer
DF27	0B	dec	bc	diminuer le compteur
DF28	79	ld	a,c	
DF29	B0	or	b	
DF2A	C0	ret	nz	
DF2B	1E17	ld	e,17	'Line too long'
DF2D	C394CA	Jp	CA94	sortir message d'erreur

***** tokens spéciaux

DF30	8C	db	8C	'DATA'
DF31	8E	db	8E	'DEFINT'
DF32	90	db	90	'DEFSTR'
DF33	8F	db	8F	'DEFREAL'
DF34	00	db	00	fin de table

DF35	CD25DF	call	DF25	écrire dans le buffer
DF38	7E	ld	a,(hl)	
DF39	B7	or	a	
DF3A	C8	ret	z	
DF3B	FE3A	cp	3A	','
DF3D	280A	Jr	z,DF49	
DF3F	23	inc	hl	
DF40	FE22	cp	22	''
DF42	20F1	Jr	nz,DF35	
DF44	CDBFEO	call	EOBF	
DF47	18EF	Jr	DF38	

BASIC 1.0

DF49	AF	xor	a	
DF4A	3239AE	ld	(AE39),a	
DF4D	C9	ret		
DF4E	C5	push	bc	
DF4F	D5	push	de	
DF50	E5	push	hl	
DF51	CD16AC	call	AC16	ret
DF54	7E	ld	a,(hl)	
DF55	23	inc	hl	
DF56	CD8AFF	call	FF8A	convertir minuscules en majuscules
DF59	CDDDE2	call	E2DD	calculer adresse des mots instruction
DF5C	CD27E3	call	E327	
DF5F	3028	jr	nc,DF89	
DF61	79	ld	a,c	
DF62	E67F	and	7F	
DF64	CD7BFF	call	FF7B	tester si lettre ou chiffre
DF67	300B	jr	nc,DF74	
DF69	1A	ld	a,(de)	
DF6A	FEE4	cp	E4	'FN'
DF6C	2806	jr	z,DF74	
DF6E	7E	ld	a,(hl)	
DF6F	CD7BFF	call	FF7B	tester si lettre ou chiffre
DF72	3815	jr	c,DF89	
DF74	F1	pop	af	
DF75	1A	ld	a,(de)	
DF76	B7	or	a	
DF77	FAC8DF	jp	m,DFC8	
DF7A	D1	pop	de	
DF7B	C1	pop	bc	
DF7C	F5	push	af	
DF7D	3EFF	ld	a,FF	'fonction'
DF7F	CD25DF	call	DF25	écrire dans buffer
DF82	F1	pop	af	
DF83	CD25DF	call	DF25	écrire dans buffer
DF86	AF	xor	a	
DF87	183A	jr	DFC3	
DF89	E1	pop	hl	

BASIC 1.0

DF8A	D1	pop	de	
DF8B	C1	pop	bc	
DF8C	E5	push	hl	
DF8D	2B	dec	hl	
DF8E	23	inc	hl	
DF8F	7E	ld	a,(hl)	
DF90	CD7BFF	call	FF7B	tester si lettre ou chiffre
DF93	38F9	jr	c,DF8E	
DF95	CDEADF	call	DFEA	
DF98	3804	jr	c,DF9E	
DF9A	3E0D	ld	a,0D	token pour variable
DF9C	1806	jr	DFA4	
DF9E	23	inc	hl	
DF9F	FE05	cp	05	
DFA1	2001	jr	nz,DFA4	
DFA3	3D	dec	a	
DFA4	CD25DF	call	DF25	écrire dans buffer
DFA7	AF	xor	a	zéro
DFA8	CD25DF	call	DF25	écrire dans buffer
DFAB	AF	xor	a	
DFAC	CD25DF	call	DF25	écrire dans buffer
DFAF	E3	ex	(sp),hl	
DFB0	7E	ld	a,(hl)	
DFB1	CD7BFF	call	FF7B	tester si lettre ou chiffre
DFB4	3007	jr	nc,DFBD	
DFB6	7E	ld	a,(hl)	
DFB7	CD25DF	call	DF25	écrire dans buffer
DFBA	23	inc	hl	
DFBB	18F3	jr	DFB0	
DFBD	CDDFE0	call	EODF	
DFC0	E1	pop	hl	
DFC1	3EFF	ld	a,FF	
DFC3	3239AE	ld	(AE39),a	
DFC6	37	scf		
DFC7	C9	ret		
DFC8	E5	push	hl	
DFC9	4F	ld	c,a	

BASIC 1.0

DFCA	21DCDF	ld	hl,DFDC	adresse de base de la table
DFCD	CDAAFF	call	FFAA	parcourir la table
DFD0	9F	sbc	a,a	
DFD1	E601	and	01	
DFD3	3239AE	ld	(AE39),a	
DFD6	79	ld	a,c	
DFD7	E1	pop	hl	
DFD8	D1	pop	de	
DFD9	C1	pop	bc	
DFDA	B7	or	a	
DFDB	C9	ret		

***** instructions avec numéro de ligne

DFC	C7	db	C7	'RESTORE'
DFDD	81	db	81	'AUTO'
DFDE	C6	db	C6	'RENUM'
DFDF	92	db	92	'DELETE'
DFE0	96	db	96	'EDIT'
DFE1	C8	db	C8	'RESUME'
DFE2	E3	db	E3	'ERL'
DFE3	97	db	97	'ELSE'
DFE4	CA	db	CA	'RUN'
DFE5	A7	db	A7	'LIST'
DFE6	A0	db	A0	'GOTO'
DFE7	EB	db	EB	'THEN'
DFE8	9F	db	9F	'GOSUB'
DFE9	00	db	00	fin de la table

DFEA	FE26	cp	26	'&'
DFEC	D0	ret	nc	
DFED	FE21	cp	21	' '
DFE0	D0	ret	nc	
DFF1	FE22	cp	22	''''
DFF3	C8	ret	z	
DFF4	FE23	cp	23	'#'
DFF6	C8	ret	z	
DFF7	EE27	xor	27	''''
DFF9	FE04	cp	04	
DFFB	CEFF	adc	a,FF	

BASIC 1.0

DFFD	37	scf		
DFFE	c9	ret		
E000	39	add	hl,sp	
E001	AE	xor	(hl)	
E002	B7	or	a	
E003	2815	jr	z,E01A	
E005	7E	ld	a,(hl)	
E006	23	inc	hl	
E007	FA25DF	jp	m,DF25	écrire dans buffer
E00A	FE2E	cp	2E	','
E00C	CA25DF	jp	z,DF25	écrire dans buffer
E00F	2B	dec	hl	
E010	D5	push	de	
E011	CD04EE	call	EE04	
E014	3034	jr	nc,E04A	
E016	3E1E	ld	a,1E	token pour numéro de ligne
E018	184F	jr	E069	
E01A	D5	push	de	
E01B	C5	push	bc	
E01C	CDBEEC	call	ECBE	
E01F	C1	pop	bc	
E020	3028	jr	nc,E04A	
E022	CD27FF	call	FF27	tester si chaîne
E025	3E1F	ld	a,1F	token pour virgule flottante
E027	3040	jr	nc,E069	
E029	EB	ex	de,hl	
E02A	2AC2B0	ld	hl,(BOC2)	
E02D	EB	ex	de,hl	
E02E	7A	ld	a,d	
E02F	B7	or	a	
E030	3E1A	ld	a,1A	token pour nombre deux octets
E032	2035	jr	nz,E069	
E034	E3	ex	(sp),hl	
E035	EB	ex	de,hl	
E036	7D	ld	a,l	
E037	FE0A	cp	0A	10
E039	3004	jr	nc,E03F	
E03B	C60E	add	a,0E	additionner offset

BASIC 1.0

E03D	1806	jr	E045	
E03F	3E19	ld	a,19	token pour valeur sur un octet
E041	CD25DF	call	DF25	écrire dans buffer
E044	7D	ld	a,1	
E045	CD25DF	call	DF25	écrire dans buffer
E048	E1	pop	hl	
E049	C9	ret		
E04A	7E	ld	a,(hl)	
E04B	23	inc	hl	
E04C	E3	ex	(sp),hl	
E04D	EB	ex	de,hl	
E04E	CD25DF	call	DF25	écrire dans buffer
E051	EB	ex	de,hl	
E052	E3	ex	(sp),hl	
E053	CDB8FF	call	FFB8	comparer hl <> de
E056	20F2	jr	nz,E04A	
E058	D1	pop	de	
E059	C9	ret		
E05A	D5	push	de	
E05B	C5	push	bc	
E05C	CDBEEC	call	ECBE	
E05F	C1	pop	bc	
E060	30E8	jr	nc,E04A	
E062	FE02	cp	02	
E064	3E1B	ld	a,1B	token pour nombre binaire
E066	2801	jr	z,E069	
E068	3C	inc	a	
E069	D1	pop	de	
E06A	CD25DF	call	DF25	écrire dans buffer
E06D	E5	push	hl	
E06E	21C2B0	ld	hl,B0C2	
E071	CD23FF	call	FF23	aller chercher type de variable
E074	F5	push	af	
E075	7E	ld	a,(hl)	
E076	23	inc	hl	
E077	CD25DF	call	DF25	écrire dans buffer
E07A	F1	pop	af	

BASIC 1.0

E07B	3D	dec	a	
E07C	20F6	jr	nz,E074	
E07E	E1	pop	hl	
E07F	C9	ret		
E080	FE22	cp	22	'''
E082	283B	jr	z,E0BF	
E084	FE7C	cp	7C	'EBIEA', extension d'instruction
E086	2845	jr	z,E0CD	
E088	C5	push	bc	
E089	D5	push	de	
E08A	EE3F	xor	3F	'?'
E08C	06BF	ld	b,BF	'PRINT'
E08E	2816	jr	z,E0A6	
E090	2B	dec	hl	
E091	114BE6	ld	de,E64B	, adresse des opérateurs Basic
E094	CD27E3	call	E327	
E097	1A	ld	a,(de)	
E098	3808	jr	c,E0A2	
E09A	7E	ld	a,(hl)	
E09B	FE20	cp	20	'5'
E09D	3002	jr	nc,E0A1	
E09F	3E20	ld	a,20	'5'
E0A1	23	inc	hl	
E0A2	47	ld	b,a	
E0A3	CDB3E0	call	E0B3	
E0A6	3239AE	ld	(AE39),a	
E0A9	78	ld	a,b	
E0AA	D1	pop	de	
E0AB	C1	pop	bc	
E0AC	FEC0	cp	C0	'''
E0AE	2836	jr	z,E0E6	
E0B0	C325DF	jp	DF25	écrire dans buffer
E0B3	3D	dec	a	
E0B4	C8	ret	z	
E0B5	EE22	xor	22	'''
E0B7	C8	ret	z	
E0B8	3A39AE	ld	a,(AE39)	
E0BB	3C	inc	a	

BASIC 1.0

EOBC	C8	ret	z	
EOBD	3D	dec	a	
EOBE	C9	ret		
EOBF	CD25DF	call	DF25	écrire dans buffer
EOC2	7E	ld	a, (h1)	
EOC3	B7	or	a	
EOC4	C8	ret	z	
EOC5	23	inc	h1	
EOC6	FE22	cp	22	'''
EOC8	20F5	Jr	nz,EOBF	
EOCA	C325DF	jp	DF25	écrire dans buffer
*****				traiter extension d'instruction
EOCD	CD25DF	call	DF25	écrire dans buffer
EOD0	AF	xor	a	zéro

BASIC 1.0

EOD1	3239AE	ld	(AE39),a	
EOD4	CD25DF	call	DF25	écrire dans buffer
EOD7	7E	ld	a,(hl)	prochain caractère
EOD8	23	inc	hl	augmenter pointeur
EOD9	CD7BFF	call	FF7B	tester si lettre ou chiffre
EODC	38F6	Jr	c,EOD4	oui, alors dans buffer
EODE	2B	dec	hl	
EODF	1B	dec	de	rétrograder pointeur
EOE0	1A	ld	a,(de)	
EOE1	F680	or	80	mettre bit 7 pour dernier caractère
EOE3	12	ld	(de),a	
EOE4	13	inc	de	
EOE5	C9	ret		

EOE6	3E01	ld	a,01	
EOE8	CD25DF	call	DF25	écrire dans buffer
EOEB	3ECO	ld	a,C0	'''
EOED	CD25DF	call	DF25	écrire dans buffer
EOF0	7E	ld	a,(hl)	caractère
EOF1	23	inc	hl	
EOF2	B7	or	a	jusqu'à fin de ligne
EOF3	20F8	Jr	nz,EOED	écrire dans buffer
EOF5	2B	dec	hl	
EOF6	C9	ret		

*****				instruction Basic LIST
EOF7	CDBOCE	call	CEB0	aller chercher zone de numéros de ligne
EOFA	C5	push	bc	
EOFB	D5	push	de	
EOFC	CDC6C1	call	C1C6	aller chercher numéro canal
EOFF	CD4ADD	call	DD4A	fin de l'instruction, sinon 'Syntax error'
E102	CDCBDD	call	DDCB	adresse de ligne actuelle sur zéro
E105	D1	pop	de	
E106	C1	pop	bc	
E107	CDODE1	call	E10D	lister lignes
E10A	C364C0	Jp	C064	au mode READY

***** lister lignes Basic bc -de

BASIC 1.0

E10D	D5	push	de	
E10E	50	ld	d,b	numéro de ligne dans de
E10F	59	ld	e,c	
E110	CDA3E7	call	E7A3	chercher ligne Basic de
E113	D1	pop	de	
E114	4E	ld	c,(hl)	
E115	23	inc	hl	fin du programme ?
E116	46	ld	b,(hl)	
E117	2B	dec	hl	
E118	78	ld	a,b	
E119	B1	or	c	
E11A	C8	ret	z	terminé
E11B	CD3CC4	call	C43C	interruption par 'ESC' ?
E11E	E5	push	hl	
E11F	09	add	hl,bc	additionner longueur de ligne
E120	E3	ex	(sp),hl	
E121	D5	push	de	
E122	E5	push	hl	
E123	23	inc	hl	
E124	23	inc	hl	
E125	5E	ld	e,(hl)	
E126	23	inc	hl	prochain numéro de ligne dans de
E127	56	ld	d,(hl)	
E128	E1	pop	hl	
E129	E3	ex	(sp),hl	
E12A	CDB8FF	call	FFB8	comparer hl <> de
E12D	E3	ex	(sp),hl	
E12E	3812	jr	c,E142	supérieur dernier numéro de ligne ?
E130	CD63E1	call	E163	lister ligne Basic dans buffer
E133	CD45E1	call	E145	sortir un caractère tiré du buffer
E136	23	inc	hl	
E137	7E	ld	a,(hl)	
E138	B7	or	a	dernier caractère ?
E139	20F8	jr	nz,E133	non
E13B	CD4EC3	call	C34E	sortir LF
E13E	D1	pop	de	
E13F	E1	pop	hl	
E140	18D2	jr	E114	
E142	E1	pop	hl	

BASIC 1.0

E143 E1 pop hl
E144 C9 ret

E145 CDBAC1 call C1BA sortir un caractère tiré du buffer
E148 380B jr c,E155 canal de sortie inférieur 8 ?
E14A 7E ld a,(hl) oui, sortie sur écran
E14B CD6EC3 call C36E sortir caractère
E14E FE0A cp 0A LF ?
E150 C0 ret nz
E151 3E0D ld a,0D envoyer CR à la suite
E153 180B jr E160

E155 7E ld a,(hl) sortie sur écran
E156 FE20 cp 20 aller chercher caractère
E158 3006 jr nc,E160 caractère de contrôle ?
E15A 3E01 ld a,01 non, sortir tel quel
 caractères de contrôle comme
 caractères imprimables
E15C CD6EC3 call C36E sortir caractère
E15F 7E ld a,(hl) aller chercher caractère
E160 C36EC3 jp C36E et le sortir

E163 D5 push de lister ligne Basic dans buffer
E164 01A4AC ld bc,ACA4 pointeur sur buffer d'entrée
E167 C5 push bc
E168 23 inc hl
E169 23 inc hl
E16A 5E ld e,(hl)
E16B 23 inc hl numéro de ligne dans de
E16C 56 ld d,(hl)
E16D 23 inc hl
E16E E5 push hl
E16F EB ex de,hl
E170 CD0DFF call FF0D prendre nombre entier hl
E173 CD82EE call EE82 convertir en ASCII
E176 110000 ld de,0000
E179 7E ld a,(hl)
E17A 23 inc hl

BASIC 1.0

E17B	B7	or	a	
E17C	2805	jr	z,E183	
E17E	CDFEE1	call	E1FE	écrire dans buffer
E181	18F6	jr	E179	
E183	3E20	ld	a,20	'5'
E185	CDFEE1	call	E1FE	écrire dans buffer
E188	E1	pop	hl	
E189	7E	ld	a,(hl)	aller chercher caractère dans programme
E18A	B7	or	a	
E18B	2805	jr	z,E192	fin de ligne ?
E18D	CD96E1	call	E196	étendre token
E190	18F7	jr	E189	
E192	02	ld	(bc),a	
E193	E1	pop	hl	
E194	D1	pop	de	
E195	C9	ret		
E196	CD13AC	call	AC13	ret
E199	FA20E2	jp	m,E220	token d'instruction ?
E19C	FE02	cp	02	
E19E	381D	jr	c,E1BD	
E1A0	FE05	cp	05	
E1A2	3843	jr	c,E1E7	
E1A4	FE0B	cp	0B	
E1A6	3822	jr	c,E1CA	
E1A8	FE0E	cp	0E	
E1AA	383B	jr	c,E1E7	
E1AC	FE20	cp	20	'5'
E1AE	382E	jr	c,E1DE	sortir constante
E1B0	FE7C	cp	7C	'EB1EA', extension d'instruction
E1B2	2851	jr	z,E205	
E1B4	CDEADF	call	DFEA	
E1B7	DC1AE2	call	c,E21A	sortir espace
E1BA	7E	ld	a,(hl)	
E1BB	180D	jr	E1CA	
E1BD	23	inc	hl	

BASIC 1.0

E1BE	7E	ld	a,(h1)	
E1BF	FEC0	cp	C0	'''
E1C1	285D	Jr	z,E220	
E1C3	FE97	cp	97	'ELSE'
E1C5	2859	Jr	z,E220	
E1C7	2B	dec	h1	
E1C8	3E3A	ld	a,3A	':'
E1CA	1E00	ld	e,00	
E1CC	FE22	cp	22	'''
E1CE	200B	Jr	nz,E1DB	
E1D0	CDFEE1	call	E1FE	écrire dans buffer
E1D3	23	inc	h1	
E1D4	7E	ld	a,(h1)	
E1D5	B7	or	a	
E1D6	C8	ret	z	
E1D7	FE22	cp	22	'''
E1D9	20F5	Jr	nz,E1D0	
E1DB	23	inc	h1	
E1DC	1820	Jr	E1FE	écrire dans buffer
E1DE	CD1AE2	call	E21A	sortir espace
E1E1	CD53E2	call	E253	sortir constante
E1E4	1E01	ld	e,01	
E1E6	C9	ret		
E1E7	CD1AE2	call	E21A	sortir espace
E1EA	7E	ld	a,(h1)	
E1EB	F5	push	af	
E1EC	23	inc	h1	
E1ED	23	inc	h1	
E1EE	23	inc	h1	
E1EF	CD0FE2	call	E20F	
E1F2	F1	pop	af	
E1F3	1E01	ld	e,01	
E1F5	FE0B	cp	0B	
E1F7	D0	ret	nc	
E1F8	1E00	ld	e,00	
E1FA	EE27	xor	27	
E1FC	E6FD	and	FD	
E1FE	02	ld	(bc),a	écrire caractère dans buffer

BASIC 1.0

E1FF	03	inc	bc	augmenter pointeur de buffer
E200	15	dec	d	
E201	C0	ret	nz	
E202	0B	dec	bc	
E203	14	inc	d	
E204	C9	ret		
*****				lister extension d'instruction
E205	1E01	ld	e,01	
E207	CDFEE1	call	E1FE	écrire dans buffer
E20A	23	inc	hl	
E20B	7E	ld	a,(hl)	prochain caractère
E20C	23	inc	hl	augmenter pointeur
E20D	B7	or	a	
E20E	C0	ret	nz	fin de ligne ?
E20F	7E	ld	a,(hl)	aller chercher caractère
E210	E67F	and	7F	annuler bit 7
E212	CDFEE1	call	E1FE	écrire dans buffer
E215	BE	cp	(hl)	dernier caractère ?
E216	23	inc	hl	
E217	30F6	jr	nc,E20F	non, prochain caractère
E219	C9	ret		
E21A	1D	dec	e	
E21B	C0	ret	nz	
E21C	3E20	ld	a,20	'5'
E21E	18DE	jr	E1FE	écrire dans buffer
E220	23	inc	hl	
E221	FEFF	cp	FF	fonction ?
E223	2002	jr	nz,E227	non
E225	7E	ld	a,(hl)	
E226	23	inc	hl	
E227	F5	push	af	
E228	E5	push	hl	
E229	CDEDE2	call	E2ED	lister token ?
E22C	B7	or	a	
E22D	2808	jr	z,E237	
E22F	F5	push	af	
E230	CD1AE2	call	E21A	
E233	F1	pop	af	

BASIC 1.0

E234	CDFEE1	call	E1FE	écrire dans buffer
E237	7E	ld	a,(h1)	
E238	E67F	and	7F	
E23A	FE09	cp	09	
E23C	C4FEE1	call	nz,E1FE	écrire dans buffer
E23F	BE	cp	(h1)	
E240	23	inc	h1	
E241	28F4	jr	z,E237	
E243	CD7BFF	call	FF7B	tester si lettre ou chiffre
E246	1E00	ld	e,00	
E248	3002	jr	nc,E24C	
E24A	1E01	ld	e,01	
E24C	E1	pop	h1	
E24D	F1	pop	af	
E24E	D6E4	sub	E4	
E250	C0	ret	nz	
E251	5F	ld	e,a	
E252	C9	ret		
E253	D5	push	de	
E254	7E	ld	a,(h1)	
E255	23	inc	h1	
E256	FE1B	cp	1B	nombre binaire ?
E258	2849	jr	z,E2A3	
E25A	FE1C	cp	1C	nombre hexadécimal ?
E25C	2850	jr	z,E2AE	
E25E	FE1E	cp	1E	adresse de ligne ?
E260	2826	jr	z,E288	
E262	FE1D	cp	1D	numéro de ligne ?
E264	2822	jr	z,E288	
E266	FE1F	cp	1F	nombre à virgule flottante ?
E268	285E	jr	z,E2C8	
E26A	FE19	cp	19	nombre sur un octet ?
E26C	2809	jr	z,E277	
E26E	FE1A	cp	1A	nombre sur deux octets ?
E270	280B	jr	z,E27D	
E272	D60E	sub	0E	chiffre ?
E274	5F	ld	e,a	
E275	1802	jr	E279	

BASIC 1.0

```
*****
E277 5E      ld      e,(hl)   sortir nombre sur un octet
                                Lo-byte
E278 23      inc      hl
E279 1600    ld      d,00      Hi-Byte zéro
E27B 1804    jr      E281
```

```
*****
E27D 5E      ld      e,(hl)   sortir nombre sur deux octets
                                Lo-Byte
E27E 23      inc      hl
E27F 56      ld      d,(hl)   Hi-Byte
E280 23      inc      hl
E281 E3      ex      (sp),hl
E282 EB      ex      de,hl
E283 CD0DFF  call   FF0D      prendre nombre entier hl
E286 1847    jr      E2CF
```

```
*****
E288 5E      ld      e,(hl)   sortir numéro de ligne
E289 23      inc      hl
E28A 56      ld      d,(hl)
E28B 23      inc      hl
E28C FE1E    cp      1E      numéro de ligne ?
E28E 2809    jr      z,E299    oui
E290 E5      push   hl
E291 EB      ex      de,hl
E292 23      inc      hl
E293 23      inc      hl
E294 23      inc      hl
E295 5E      ld      e,(hl)
E296 23      inc      hl
E297 56      ld      d,(hl)
E298 E1      pop    hl
E299 E3      ex      (sp),hl
E29A EB      ex      de,hl
E29B CD0DFF  call   FF0D      prendre nombre entier hl
E29E CD82EE  call   EE82      convertir en ASCII
E2A1 182F    jr      E2D2
```

```
*****
E2A3 C5      push   bc      sortir nombre binaire
```

BASIC 1.0

E2A4	010200	ld	bc,0002	
E2A7	CD14F1	call	F114	convertir en nombre binaire
E2AA	3E58	ld	a,58	'X'
E2AC	1809	Jr	E2B7	

***** sortir nombre hexa

E2AE	C5	push	bc	
E2AF	010200	ld	bc,0002	
E2B2	CD19F1	call	F119	convertir en nombre hexa
E2B5	3E48	ld	a,48	'H'
E2B7	C1	pop	bc	
E2B8	E3	ex	(sp),h1	
E2B9	EB	ex	de,h1	
E2BA	F5	push	af	
E2BB	3E26	ld	a,26	'&'
E2BD	CDFEE1	call	E1FE	écrire dans buffer
E2C0	F1	pop	af	
E2C1	FE48	cp	48	'H' pas
E2C3	C4FEE1	call	nz,E1FE	écrire dans buffer
E2C6	180A	Jr	E2D2	

***** sortir nombre à virgule flottante

E2C8	3E05	ld	a,05	type de variable 'Real'
E2CA	CD4BFF	call	FF4B	aller chercher nombre
E2CD	E3	ex	(sp),h1	
E2CE	EB	ex	de,h1	
E2CF	CD8FEE	call	EE8F	convertir en ASCII
E2D2	7E	ld	a,(h1)	aller chercher caractère
E2D3	23	inc	h1	
E2D4	CDFEE1	call	E1FE	écrire dans buffer
E2D7	7E	ld	a,(h1)	
E2D8	B7	or	a	fin du nombre ?
E2D9	20F7	Jr	nz,E2D2	non
E2DB	E1	pop	h1	
E2DC	C9	ret		

E2DD	E5	push	h1	
E2DE	D641	sub	41	'A'
E2E0	87	add	a,a	

BASIC 1.0

E2E1	C654	add	a,54	
E2E3	6F	ld	l,a	plus E354, adresses des mots- Instructions
E2E4	CEE3	adc	a,E3	
E2E6	95	sub	l	
E2E7	67	ld	h,a	
E2E8	5E	ld	e,(hl)	
E2E9	23	inc	hl	
E2EA	56	ld	d,(hl)	
E2EB	E1	pop	hl	
E2EC	C9	ret		
E2ED	C5	push	bc	
E2EE	4F	ld	c,a	
E2EF	061A	ld	b,1A	26 lettres
E2F1	2188E3	ld	hl,E388	table des mots instruction
E2F4	CD13E3	call	E313	
E2F7	380D	jr	c,E306	
E2F9	23	inc	hl	
E2FA	10F8	djnz	E2F4	lettre suivante
E2FC	214BE6	ld	hl,E64B	table des opérateurs Basic
E2FF	CD13E3	call	E313	
E302	3007	jr	nc,E30B	
E304	06C0	ld	b,C0	
E306	78	d	a,b	
E307	C640	add	a,40	
E309	C1	pop	bc	
E30A	C9	ret		
E30B	CD19AC	call	AC19	ret
E30E	1E02	ld	e,02	'Syntax error'
E310	C394CA	jp	CA94	sortir message d'erreur
E313	7E	ld	a,(hl)	
E314	B7	or	a	
E315	C8	ret	z	
E316	E5	push	hl	
E317	7E	ld	a,(hl)	
E318	23	inc	hl	
E319	17	rla		

BASIC 1.0

E31A	30FB	Jr	nc,E317	
E31C	7E	ld	a,(hl)	
E31D	23	inc	hl	
E31E	B9	cp	c	
E31F	2803	Jr	z,E324	
E321	F1	pop	af	
E322	18EF	Jr	E313	
E324	E1	pop	hl	
E325	37	scf		
E326	C9	ret		
E327	1A	ld	a,(de)	
E328	B7	or	a	
E329	C8	ret	z	
E32A	E5	push	hl	
E32B	1A	ld	a,(de)	
E32C	13	inc	de	
E32D	FE09	cp	09	TAB ?
E32F	2804	Jr	z,E335	
E331	FE20	cp	20	'5'
E333	2005	Jr	nz,E33A	
E335	CD61DD	call	DD61	ignorer espace, TAB et LF
E338	18F1	Jr	E32B	
E33A	4F	ld	c,a	
E33B	7E	ld	a,(hl)	
E33C	23	inc	hl	
E33D	CD8AFF	call	FF8A	convertir minuscules en majuscules
E340	A9	xor	c	
E341	28E8	Jr	z,E32B	
E343	E67F	and	7F	
E345	280A	Jr	z,E351	
E347	1B	dec	de	
E348	1A	ld	a,(de)	
E349	13	inc	de	
E34A	17	rla		
E34B	30FB	Jr	nc,E348	
E34D	13	inc	de	
E34E	E1	pop	hl	

BASIC 1.0

```

E34F 18D6    Jr    E327
E351 F1      pop    af
E352 37      scf
E353 C9      ret
    
```

```

*****
adresses des mots instructions
E554 35E6    dw    E635    A
E356 2AE6    dw    E62A    B
E358 EFE5    dw    E5EF    C
E35A B9E5    dw    E5B9    D
E35C 8AE5    dw    E58A    E
E35E 7EE5    dw    E57E    F
E360 72E5    dw    E572    G
E362 68E5    dw    E568    H
E364 47E5    dw    E547    I
E366 43E5    dw    E543    J
E368 3FE5    dw    E53F    K
E36A 13E5    dw    E513    L
E36C EDE4    dw    E4ED    M
E36E E2E4    dw    E4E2    N
E370 AAE4    dw    E4AA    O
E372 86E4    dw    E486    P
E374 85E4    dw    E485    Q
E376 3BE4    dw    E43B    R
E378 FBE3    dw    E3FB    S
E37A CFE3    dw    E3CF    T
E37C C0E3    dw    E3C0    U
E37E B8E3    dw    E3B8    V
E380 9AE3    dw    E39A    W
E382 92E3    dw    E392    X
E384 8DE3    dw    E38D    Y
E386 88E3    dw    E388    Z
    
```

***** Table des Instructions Basic

BASIC 1.0

*****		Z
E388 4F 4E C5	DA ZONE	
E38C 00		
*****		Y
E38D 50 4F D3	48 YPOS	
E391 00		
*****		X
E392 50 4F D3	47 XPOS	
E396 4F D2	FD XOR	
E399 00		
*****		W
E39A 52 49 54 C5	D9 WRITE	
E39F 49 4E 44 4F D7	D8 WINDOW	
E3A5 49 44 54 C8	D7 WIDTH	
E3AA 48 49 4C C5	D6 WHILE	
E3AF 45 4E C4	D5 WEND	
E3B3 41 49 D4	D4 WAIT	
E3B7 00		
*****		V
E3B8 50 4F D3	7F VPOS	
E3BC 41 CC	1D VAL	
E3BF 00		
*****		U
E3C0 53 49 4E C7	ED USING	
E3C5 50 50 45 52 A4	1C UPPER\$	
E3CB 4E D4	1B UNT	
E3CE 00		
*****		T
E3CF 52 4F CE	D3 TRON	
E3D3 52 4F 46 C6	D2 TROFF	
E3D8 CF	EC TO	
E3DA 49 4D C5	46 TIME	
E3DE 48 45 CE	EB THEN	
E3E2 45 53 54 D2	7D TESTR	
E3E7 45 53 D4	7C TEST	
E3EB 41 CE	1A TAN	
E3EE 41 47 4F 46 C6	D1 TAGOFF	
E3F4 41 C7	D0 TAG	
E3F7 41 C2	EA TAB	
E3FA 00		
*****		S
E3FB 59 4D 42 4F CC	CF SYMBOL	
E401 57 41 D0	E7 SWAP	
E405 54 52 49 4E 47 A4	7B STRING\$	
E40C 54 52 A4	19 STR\$	
E410 54 4F D0	CE STOP	
E414 54 45 D0	E6 STEP	
E418 51 D2	18 SQR	

BASIC 1.0

E41B	D1	17 SQ
E41D	50 45 45 C4	CD SPEED
E422	50 C3	E5 SPC
E425	50 41 43 45 A4	16 SPACES
E42B	4F 55 4E C4	CC SOUND
E430	49 CE	15 SIN
E433	47 CE	14 SGN
E436	41 56 C5	CB SAVE
E43A	00	

*****			R
E43B	55 CE	CA RUN	
E43E	4F 55 4E C4	7A ROUND	
E443	4E C4	45 RND	
E446	49 47 48 54 A4	79 RIGHTS	
E44C	45 54 55 52 CE	C9 RETURN	
E452	45 53 55 4D C5	C8 RESUME	
E458	45 53 54 4F 52 C5	C7 RESTORE	
E45F	45 4E 55 CD	C6 RENUM	
E464	45 4D 41 49 CE	13 REMAIN	
E46A	45 CD	C5 REM	
E46D	45 4C 45 41 53 C5	C4 RELEASE	
E474	45 41 C4	C3 READ	
E478	41 4E 44 4F 4D 49 5A C5	C2 RANDOMIZE	
E481	41 C4	C1 RAD	
E484	00		

*****			Q
E485	00		

*****			P
E486	52 49 4E D4	BF PRINT	
E48B	4F D3	78 POS	
E48E	4F 4B C5	BE POKE	
E492	4C 4F 54 D2	BD PLOTR	
E497	4C 4F D4	BC PLOT	
E49B	C9	44 PI	
E49D	45 CE	BB PEN	
E4A0	45 45 CB	12 PEEK	
E4A4	41 50 45 D2	BA PAPER	
E4A9	00		

*****			O
E4AA	55 D4	B9 OUT	
E4AD	52 49 47 49 CE	B8 ORIGIN	
E4B3	D2	FC OR	
E4B5	50 45 4E 4F 55 D4	B7 OPENOUT	
E4BC	50 45 4E 49 CE	B6 OPENIN	
E4C2	4E 20 53 D1	B5 ON SQ	
E4C7	4E 20 45 52 52 4F 52 20		
E4CF	47 4F 09 54 4F 20 B0	B4 ON ERROR GO TO 0	
E4D7	4E 20 42 52 45 41 CB	B3 ON BREAK	
E4DF	CE	B2 ON	
E4E1	00		

BASIC 1.0

```
*****
E4E2 4F D4                FE NOT
E4E5 45 D7                B1 NEW
E4E8 45 58 D4            B0 NEXT
E4EC 00
```

N

```
*****
E4ED 4F 56 45 D2        AF MOVER
E4F2 4F 56 C5           AE MOVE
E4F6 4F 44 C5           AD MODE
E4FA 4F C4              FB MOD
E4FD 49 CE              77 MIN
E500 49 44 A4           AC MID$
E504 45 52 47 C5       AB MERGE
E509 45 4D 4F 52 D9    AA MEMORY
E50F 41 D8              76 MAX
E512 00
```

M

```
*****
E513 4F 57 45 52 A4    11 LOWERS
E519 4F 47 31 B0       10 LOG10
E51E 4F C7              0F LOG
E521 4F 43 41 54 C5    A9 LOCATE
E527 4F 41 C4           A8 LOAD
E52B 49 53 D4           A7 LIST
E52F 49 4E C5           A6 LINE
E533 45 D4              A5 LET
E536 45 CE              0E LEN
E539 45 46 54 A4       75 LEFT$
E53E 00
```

L

```
*****
E53F 45 D9              A4 KEY
E542 00
```

K

```
*****
E543 4F D9              0D JOY
E546 00
```

J

```
*****
E547 4E D4              0C INT
E54A 4E 53 54 D2       74 INSTR
E54F 4E 50 55 D4       A3 INPUT
E554 4E D0              0B INP
E557 4E 4B 45 59 A4    43 INKEY$
E55D 4E 4B 45 D9       0A INKEY
E562 4E CB              A2 INK
E565 C6                 A1 IF
E567 00
```

I

```
*****
E568 49 4D 45 CD       42 HIMEM
E56D 45 58 A4          73 HEX$
E571 00
```

H

BASIC 1.0

```

*****
E572 4F 09 54 CF          A0 GO TO
E577 4F 09 53 55 C2      9F GO SUB
E57D 00

```

G

```

*****
E57E 52 C5                09 FRE
E581 4F D2                9E FOR
E584 CE                   E4 FN
E586 49 D8                08 FIX
E589 00

```

F

```

*****
E58A 58 D0                07 EXP
E58D 56 45 52 D9         9D EVERY
E592 52 52 4F D2        9C ERROR
E597 52 D2                41 ERR
E59A 52 CC                E3 ERL
E59D 52 41 53 C5         9B ERASE
E5A2 4F C6                40 EOF
E5A5 4E D6                9A ENV
E5A8 4E D4                99 ENT
E5AB 4E C4                98 END
E5AE 4C 53 C5            97 ELSE
E5B2 C9                   DC EI
E5B4 44 49 D4            96 EDIT
E5B8 00

```

E

```

*****
E5B9 52 41 57 D2         95 DRAWR
E5BE 52 41 D7             94 DRAW
E5C2 49 CD                93 DIM
E5C5 C9                   DB DI
E5C7 45 4C 45 54 C5      92 DELETE
E5CD 45 C7                91 DEG
E5D0 45 46 53 54 D2      90 DEFSTR
E5D6 45 46 52 45 41 CC   8F DEFREAL
E5DD 45 46 49 4E D4      8E DEFINT
E5E3 45 C6                8D DEF
E5E6 45 43 A4             72 DEC$
E5EA 41 54 C1             8C DATA
E5EE 00

```

D

```

*****
E5EF 52 45 41 CC          06 CREAL
E5F4 4F D3 OS 05          05 COS
E5F7 4F 4E D4             8B CONT
E5FB 4C D3                8A CLS
E5FE 4C 4F 53 45 4F 55 D4 89 CLOSEOUT
E606 4C 4F 53 45 49 CE   88 CLOSEIN
E60D 4C C7                87 CLG
E610 4C 45 41 D2          86 CLEAR
E615 49 4E D4             04 CINT
E619 48 52 A4             03 CHR$
E61D 48 41 49 CE          85 CHAIN

```

C

BASIC 1.0

E622	41	D4	84	CAT	
E625	41	4C	CC	83	CALL
E629	00				

E62A	4F	52	44	45	D2	82	BORDER
E630	49	4E	A4			71	BINS
E634	00						

B

E635	55	54	CF	81	AUTO
E639	54	CE		02	ATN
E63C	53	C3		01	ASC
E63F	4E	C4		FA	AND
E642	46	54	45	80	AFTER
E647	42	D3		00	ABS
E64A	00				

A

BASIC 1.0

***** opérateurs Basic et token correspondants

E64B	DE	db	DE	'^'
E64C	F8	db	F8	
E64D	DC	db	DC	'Backslash'
E64E	F9	db	F9	
E64F	3E09BD	db	3E,09,BD	'>='
E652	F0	db	F0	
E653	3D20BE	db	3D,20,BE	'=>'
E656	F0	db	F0	
E657	BE	db	BE	'>'
E658	EE	db	EE	
E659	BD	db	BD	'='
E65A	EF	db	EF	
E65B	3C09BE	db	3C,09,BE	'<>'
E65E	F2	db	F2	
E65F	3C09BD	db	3C,09,BD	'<='
E662	F3	db	F3	
E663	3D20BC	db	3D,20,BC	'<='
E666	F3	db	F3	
E667	BC	db	BC	'<'
E668	F1	db	F1	
E669	AF	db	AF	'/'
E66A	F7	db	F7	
E66B	BA	db	BA	':'
E66C	01	db	01	
E66D	AA	db	AA	'*'
E66E	F6	db	F6	
E66F	AD	db	AD	'-'
E670	F5	db	F5	
E671	AB	db	AB	'+'
E672	F4	db	F4	
E673	A7	db	A7	''
E674	C0	db	C0	
E675	00	db	00	

***** supprimer pointeur de programme

E676	AF	xor	a	
E677	323AAE	ld	(AE3A),a	
E67A	2A81AE	ld	h1,(AE81)	début de programme
E67D	77	ld	(h1),a	

BASIC 1.0

```

E67E 23      inc    hl
E67F 77      ld     (hl),a      trois fois zéro en fin de programme
E680 23      inc    hl
E681 77      ld     (hl),a
E682 23      inc    hl
E683 2283AE  ld     (AE83),hl    fin de programme
E686 C9      ret

```

```

E687 3A3AAE  ld     a,(AE3A)
E68A B7      or     a
E68B C8      ret    z
E68C C5      push  bc
E68D D5      push  de
E68E E5      push  hl
E68F 019DE6  ld     bc,E69D      'utiliser numéros de ligne
E692 CDFFE8  call  E8FF
E695 AF      xor    a
E696 323AAE  ld     (AE3A),a
E699 E1      pop   hl
E69A D1      pop   de
E69B C1      pop   bc
E69C C9      ret

```

```

remplacer adresse de ligne par numéros de ligne
E69D CD43E9  call  E943      aller chercher prochain élément de
la ligne
E6A0 FE02   cp    02      fin de l'instruction ?
E6A2 D8     ret    c      oui
E6A3 FE1D   cp    1D      'adresse de ligne' ?
E6A5 20F6   Jr    nz,E69D   non
E6A7 56     ld    d,(hl)
E6A8 2B     dec   hl
E6A9 5E     ld    e,(hl)
E6AA 2B     dec   hl
E6AB E5     push hl
E6AC EB     ex   de,hl
E6AD 23     inc   hl
E6AE 23     inc   hl
E6AF 23     inc   hl

```

BASIC 1.0

E6B0	5E	ld	e,(h1)	
E6B1	23	inc	h1	numéro de ligne dans de
E6B2	56	ld	d,(h1)	
E6B3	E1	pop	h1	
E6B4	361E	ld	(h1),1E	'numéro de ligne'
E6B6	23	inc	h1	
E6B7	73	ld	(h1),e	
E6B8	23	inc	h1	mettre en place
E6B9	72	ld	(h1),d	
E6BA	18E1	Jr	E69D	

***** convertir ligne d'entrée en code interpréteur

E6BC	CD61DD	call	DD61	ignorer espace, TAB et LF [berlesen
E6BF	B7	or	a	
E6C0	37	scf		
E6C1	C8	ret	z	
E6C2	CD04EE	call	EE04	
E6C5	D0	ret	nc	
E6C6	7E	ld	a,(h1)	
E6C7	FE20	cp	20	'5'
E6C9	2001	Jr	nz,E6CC	
E6CB	23	inc	h1	
E6CC	CDD2E6	call	E6D2	convertir instruction en code interpréteur
E6CF	37	scf		
E6D0	9F	sbc	a,a	
E6D1	C9	ret		

***** convertir instruction en code interpréteur

E6D2	CD87E6	call	E687	
E6D5	CDBBDE	call	DEBB	token dans buffer à partir de (&AE7F) (&40)
E6D8	E5	push	h1	
E6D9	CD61DD	call	DD61	ignorer espace, TAB et LF
E6DC	B7	or	a	
E6DD	2828	Jr	z,E707	
E6DF	C5	push	bc	
E6E0	D5	push	de	
E6E1	210400	ld	h1,0004	
E6E4	09	add	h1,bc	

BASIC 1.0

E6E5	E5	push	hl	
E6E6	E5	push	hl	
E6E7	CDA3E7	call	E7A3	chercher ligne Basic de
E6EA	E5	push	hl	
E6EB	DC0BE7	call	c,E70B	
E6EE	D1	pop	de	
E6EF	C1	pop	bc	nombre d'octets
E6F0	CDF8F5	call	F5F8	réserver place dans zone de variables
E6F3	CD2CF5	call	F52C	augmenter pointeurs prg et variable de bc
E6F6	EB	ex	de,hl	
E6F7	D1	pop	de	
E6F8	73	ld	(hl),e	
E6F9	23	inc	hl	
E6FA	72	ld	(hl),d	
E6FB	23	inc	hl	
E6FC	D1	pop	de	
E6FD	73	ld	(hl),e	
E6FE	23	inc	hl	
E6FF	72	ld	(hl),d	
E700	23	inc	hl	
E701	C1	pop	bc	
E702	EB	ex	de,hl	
E703	E1	pop	hl	
E704	C3F2FF	jp	FFF2	ldir
E707	E1	pop	hl	
E708	CD9AE7	call	E79A	chercher ligne Basic de
E70B	C5	push	bc	
E70C	E5	push	hl	
E70D	09	add	hl,bc	
E70E	EB	ex	de,hl	
E70F	2A89AE	ld	hl,(AE89)	fin des tableaux
E712	CDCFFF	call	FFCF	hl := hl - de
E715	44	ld	b,h	
E716	4D	ld	c,l	
E717	EB	ex	de,hl	
E718	D1	pop	de	
E719	78	ld	a,b	

BASIC 1.0

```

E71A B1      or      c
E71B C4F2FF  call    nz,FFF2    ldir
E71E D1      pop     de
E71F 210000  ld      hl,0000
E722 CDDAFF  call    FFDA      bc := hl - de
E725 C32CF5  jp      F52C      augmenter pointeurs prg et variable
                        de bc

```

instruction Basic DELETE

```

E728 CD37E7  call    E737
E72B CD4ADD  call    DD4A      fin de l'instruction, sinon 'Syntax
                        error'

E72E CD5AE7  call    E75A
E731 CD7AC1  call    C17A
E734 C364C0  jp      C064      au mode READY

```

E737	CDBOCE	call	CEB0	aller chercher zone numéros de ligne
E73A	E5	push	hl	
E73B	C5	push	bc	
E73C	CDC1E7	call	E7C1	chercher ligne Basic de
E73F	D1	pop	de	
E740	E5	push	hl	
E741	CDA3E7	call	E7A3	chercher ligne Basic de
E744	223BAE	ld	(AE3B),hl	
E747	EB	ex	de,hl	
E748	E1	pop	hl	
E749	CDCFFF	call	FFCF	hl := hl -de
E74C	223DAE	ld	(AE3D),hl	
E74F	3804	jr	c,E755	'Improper argument'
E751	7C	ld	a,h	
E752	B5	or	l	
E753	E1	pop	hl	
E754	C0	ret	nz	
E755	1E05	ld	e,05	'Improper argument'
E757	C394CA	jp	CA94	sortir message d'erreur

E75A	CD87E6	call	E687	remplacer adresses de ligne par numéros
E75D	ED4B3DAE	ld	bc,(AE3D)	
E761	2A3BAE	ld	hl,(AE3B)	
E764	C30BE7	jp	E70B	

E767	23	inc	hl	aller chercher adresse de ligne
E768	5E	ld	e,(hl)	
E769	23	inc	hl	numéro ou adresse dans de
E76A	56	ld	d,(hl)	
E76B	23	inc	hl	
E76C	FE1D	cp	1D	'adresse de ligne' ?
E76E	C8	ret	z	oui, terminé
E76F	FE1E	cp	1E	'numéro de ligne' ?
E771	C2EAE8	jp	nz,E8EA	non, 'Syntax error'
E774	E5	push	hl	
E775	CDD6DD	call	DDD6	aller chercher numéro ligne dans hl

BASIC 1.0

```

E778 DCB8FF  call  c,FFB8      comparer hl <> de
E77B 3009    jr     nc,E786      inférieur, alors chercher début PRG
E77D E1      pop   hl
E77E E5      push  hl
E77F CDF3E8  call  E8F3          ignorer reste de la ligne
E782 23      inc   hl            à partir de l'adresse (hl)
E783 CDA7E7  call  E7A7          chercher ligne Basic
E786 D49AE7  call  nc,E79A      pas trouvé, chercher à partir de
                        début du programme

```

```

E789 2B      dec   hl
E78A EB      ex    de,hl
E78B E1      pop   hl
E78C E5      push  hl
E78D 3E1D    ld    a,1D          'adresse de ligne'
E78F 323AAE  ld    (AE3A),a
E792 2B      dec   hl
E793 72      ld    (hl),d
E794 2B      dec   hl            placer adresse de ligne dans le PRG
E795 73      ld    (hl),e
E796 2B      dec   hl
E797 77      ld    (hl),a      token pour 'adresse de ligne'
E798 E1      pop   hl
E799 C9      ret

```

```

E79A CDA3E7  call  E7A3          chercher ligne Basic
E79D D8      ret   c            chercher ligne Basic
E79E 1E08    ld    e,08         trouvée ?
E7A0 C394CA  jp    CA94         'Line does not exist'
                        sortir message d'erreur

```

```

E7A3 2A81AE  ld    hl,(AE81)    chercher ligne Basic dans (de)
                        début du programme
E7A6 23      inc   hl
E7A7 4E      ld    c,(hl)
E7A8 23      inc   hl            longueur de ligne dans bc
E7A9 46      ld    b,(hl)
E7AA 2B      dec   hl
E7AB 78      ld    a,b
E7AC B1      or    c            fin du programme ?
E7AD C8      ret   z          pas trouvé

```

BASIC 1.0

E7AE	E5	push	hl	
E7AF	23	inc	hl	
E7B0	23	inc	hl	
E7B1	7E	ld	a,(hl)	
E7B2	23	inc	hl	numéro de ligne dans hl
E7B3	66	ld	h,(hl)	
E7B4	6F	ld	l,a	
E7B5	EB	ex	de,hl	
E7B6	CDB8FF	call	FFB8	comparer hl <> de
E7B9	EB	ex	de,hl	
E7BA	E1	pop	hl	
E7BB	3F	cgf		
E7BC	D0	ret	nc	supérieur, pas trouvé
E7BD	C8	ret	z	égal, trouvé
E7BE	09	add	hl,bc	additionner longueur de ligne
E7BF	18E6	Jr	E7A7	continuer à chercher

				chercher ligne Basic
E7C1	2A81AE	ld	hl,(AE81)	début du programme
E7C4	23	inc	hl	
E7C5	E5	push	hl	ranger adresse de ligne
E7C6	4E	ld	c,(hl)	
E7C7	23	inc	hl	longueur de ligne dans bc
E7C8	46	ld	b,(hl)	
E7C9	23	inc	hl	
E7CA	78	ld	a,b	
E7CB	B1	or	c	
E7CC	280F	Jr	z,E7DD	fin du programme ?
E7CE	7E	ld	a,(hl)	
E7CF	23	inc	hl	numéro de ligne dans hl
E7D0	66	ld	h,(hl)	
E7D1	6F	ld	l,a	
E7D2	EB	ex	de,hl	
E7D3	CDB8FF	call	FFB8	comparer hl <> de
E7D6	EB	ex	de,hl	
E7D7	3804	Jr	c,E7DD	No de ligne actuel supérieur/égal?
E7D9	E1	pop	hl	
E7DA	09	add	hl,bc	additionner longueur de ligne
E7DB	18E8	Jr	E7C5	continuer à chercher

BASIC 1.0

E7DD	E1	pop	hl	hl désigne adresse de ligne
E7DE	C9	ret		
*****				instruction Basic RENUM
E7DF	110A00	ld	de,000A	10, défaut pour valeur de départ
E7E2	2805	jr	z,E7E9	
E7E4	FE2C	cp	2C	','
E7E6	C4E1CE	call	nz.CEE1	aller chercher No de ligne dans de
E7E9	D5	push	de	
E7EA	110000	ld	de,0000	0
E7ED	CD55DD	call	DD55	virgule suit ?
E7F0	3005	jr	nc,E7F7	non
E7F2	FE2C	cp	2C	','
E7F4	C4E1CE	call	nz.CEE1	non
E7F7	D5	push	de	
E7F8	110A00	ld	de,000A	10, défaut pour incrément
E7FB	CD55DD	call	DD55	virgule suit ?
E7FE	DCE1CE	call	c,CEE1	oui, aller chercher No ligne dans de
E801	CD4ADD	call	DD4A	fin de ligne, sinon 'Syntax error'
E804	E1	pop	hl	
E805	EB	ex	de,hl	
E806	E3	ex	(sp),hl	
E807	EB	ex	de,hl	
E808	D5	push	de	
E809	E5	push	hl	
E80A	CDA3E7	call	E7A3	chercher ligne Basic
E80D	D1	pop	de	
E80E	E5	push	hl	
E80F	CDA3E7	call	E7A3	chercher ligne Basic
E812	EB	ex	de,hl	
E813	E1	pop	hl	
E814	CDB8FF	call	FFB8	comparer hl <> de
E817	DA55E7	jp	c,E755	'Improper argument'
E81A	EB	ex	de,hl	
E81B	D1	pop	de	
E81C	C1	pop	bc	
E81D	D5	push	de	
E81E	E5	push	hl	
E81F	C5	push	bc	
E820	4E	ld	c,(hl)	

BASIC 1.0

E821	23	inc	hl	
E822	46	ld	b, (hl)	
E823	78	ld	a,b	
E824	B1	or	c	
E825	2813	jr	z,E83A	
E827	2B	dec	hl	
E828	09	add	hl,bc	
E829	7E	ld	a, (hl)	
E82A	23	inc	hl	
E82B	B6	or	(hl)	
E82C	280C	jr	z,E83A	
E82E	2B	dec	hl	
E82F	C1	pop	bc	
E830	E5	push	hl	
E831	EB	ex	de,hl	
E832	09	add	hl,bc	
E833	EB	ex	de,hl	
E834	DA55E7	jp	c,E755	'Improper argument'
E837	E1	pop	hl	
E838	18E5	Jr	E81F	
E83A	0164E8	ld	bc,E864	
E83D	CDFFE8	call	E8FF	
E840	C1	pop	bc	
E841	E1	pop	hl	
E842	D1	pop	de	
E843	C5	push	bc	
E844	E5	push	hl	
E845	4E	ld	c, (hl)	
E846	23	inc	hl	
E847	46	ld	b, (hl)	
E848	23	inc	hl	
E849	78	ld	a,b	
E84A	B1	or	c	
E84B	280C	jr	z,E859	
E84D	73	ld	(hl),e	
E84E	23	inc	hl	
E84F	72	ld	(hl),d	
E850	23	inc	hl	
E851	E1	pop	hl	

BASIC 1.0

E852	09	add	hl, bc	
E853	C1	pop	bc	
E854	EB	ex	de, hl	
E855	09	add	hl, bc	
E856	EB	ex	de, hl	
E857	18EA	Jr	E843	
E859	E1	pop	hl	
E85A	E1	pop	hl	
E85B	0188E8	ld	bc, E888	
E85E	CDFFE8	call	E8FF	
E861	C364C0	Jp	C064	au mode READY
E864	CD43E9	call	E943	aller chercher prochain élément de la ligne
E867	FE02	cp	02	
E869	D8	ret	c	
E86A	FE1E	cp	1E	'numéro de ligne'
E86C	20F6	Jr	nz, E864	
E86E	E5	push	hl	
E86F	56	ld	d, (hl)	
E870	2B	dec	hl	
E871	5E	ld	e, (hl)	
E872	CDA3E7	call	E7A3	chercher ligne Basic
E875	300E	Jr	nc, E885	
E877	2B	dec	hl	
E878	EB	ex	de, hl	
E879	E1	pop	hl	
E87A	E5	push	hl	
E87B	72	ld	(hl), d	
E87C	2B	dec	hl	
E87D	73	ld	(hl), e	
E87E	2B	dec	hl	
E87F	3E1D	ld	a, 1D	'adresse de ligne'
E881	77	ld	(hl), a	
E882	323AAE	ld	(AE3A), a	
E885	E1	pop	hl	
E886	18DC	Jr	E864	
E888	CD43E9	call	E943	aller chercher prochain élément de

BASIC 1.0

```

                                la ligne
E88B FE02    cp      02
E88D D8      ret     c
E88E FE1E    cp      1E      'numéro de ligne'
E890 20F6    jr      nz,E888
E892 E5      push    hl
E893 56      ld      d,(hl)
E894 2B      dec     hl
E895 5E      ld      e,(hl)
E896 CDD6DD  call    DDD6      numéro de ligne dans hl
E899 CD18CB  call    CB18      'Undefined line in'
E89C E1      pop     hl
E89D 1809    jr      E888

E89F 0601    ld      b,01
E8A1 2B      dec     hl
E8A2 CD43E9  call    E943      aller chercher prochain élément de
                                la ligne

E8A5 B7      or      a
E8A6 C8      ret     z
E8A7 FE01    cp      01
E8A9 2807    jr      z,E8B2
E8AB FEA1    cp      A1      'IF'
E8AD 20F3    jr      nz,E8A2
E8AF 04      inc     b
E8B0 18F0    jr      E8A2

E8B2 CD43E9  call    E943      aller chercher prochain élément de
                                la ligne
E8B5 FE97    cp      97      'ELSE'
E8B7 20EC    jr      nz,E8A5
E8B9 05      dec     b
E8BA 20E6    jr      nz,E8A2
E8BC CD3FDD  call    DD3F      ignorer les espaces
E8BF 04      inc     b
E8C0 C9      ret

***** tester si variable indiquée
E8C1 7E      ld      a,(hl)
E8C2 FE5B    cp      5B      'ET'

```

BASIC 1.0

E8C4	2803	Jr	z,E8C9	
E8C6	FE28	cp	28	'('
E8C8	C0	ret	nz	pas d'index
E8C9	0600	ld	b,00	
E8CB	04	inc	b	augmenter imbrication de parenthèses
E8CC	CD43E9	call	E943	aller chercher prochain élément de la ligne
E8CF	FE5B	cp	5B	'ET'
E8D1	28F8	Jr	z,E8CB	
E8D3	FE28	cp	28	'('
E8D5	28F4	Jr	z,E8CB	
E8D7	FE5D	cp	5D	'EU'
E8D9	280A	Jr	z,E8E5	
E8DB	FE29	cp	29	'),'
E8DD	2806	Jr	z,E8E5	
E8DF	FE02	cp	02	
E8E1	3807	Jr	c,E8EA	'Syntax error'
E8E3	18E7	Jr	E8CC	
E8E5	05	dec	b	diminuer imbrication
E8E6	20E4	Jr	nz,E8CC	encore parenthèses isolées ?
E8E8	23	inc	hl	hl pointé maintenant sur après index
E8E9	C9	ret		
E8EA	1E02	ld	e,02	'Syntax error'
E8EC	C394CA	jp	CA94	sortir message d'erreur
*****				instruction Basic DATA
E8EF	0601	ld	b,01	':', fin de l'instruction
E8F1	1802	Jr	E8F5	
*****				instruction Basice ELSE, REM et '
E8F3	0600	ld	b,00	0, fin de la ligne
E8F5	2B	dec	hl	
E8F6	CD43E9	call	E943	aller chercher prochain élément de la ligne
E8F9	B7	or	a	
E8FA	C8	ret	z	
E8FB	B8	cp	b	atteint marque de fin ?
E8FC	20F8	Jr	nz,E8F6	non

BASIC 1.0

E8FE	C9	ret		
E8FF	CDD2DD	call	DDD2	adresse de ligne actuelle dans hl
E902	E5	push	hl	
E903	2A81AE	ld	hl,(AE81)	début du programme
E906	23	inc	hl	
E907	7E	ld	a,(hl)	
E908	23	inc	hl	
E909	B6	or	(hl)	
E90A	2813	Jr	z,E91F	
E90C	23	inc	hl	
E90D	CDCEDD	call	DDCE	fixer adresse de ligne actuelle
E910	23	inc	hl	
E911	C5	push	bc	
E912	CD99FF	call	FFF9	Jp (bc)
E915	C1	pop	bc	
E916	2B	dec	hl	
E917	CD35E9	call	E935	
E91A	B7	or	a	
E91B	20F4	Jr	nz,E911	
E91D	18E7	Jr	E906	
E91F	E1	pop	hl	
E920	C3CEDD	Jp	DDCE	fixer adresse de ligne actuelle
E923	CD35E9	call	E935	
E926	B7	or	a	
E927	C0	ret	nz	
E928	23	inc	hl	
E929	7E	ld	a,(hl)	
E92A	23	inc	hl	
E92B	B6	or	(hl)	
E92C	59	ld	e,c	
E92D	CA94CA	Jp	z,CA94	sortir message d'erreur
E930	23	inc	hl	
E931	54	ld	d,h	
E932	5D	ld	e,l	
E933	23	inc	hl	
E934	C9	ret		

BASIC 1.0

E935	CD43E9	call	E943	aller chercher prochain élément de la ligne
E938	FE02	cp	02	fin de ligne ?
E93A	D8	ret	c	
E93B	FE97	cp	97	'ELSE'
E93D	C8	ret	z	
E93E	FEEB	cp	EB	'THEN'
E940	20F3	Jr	nz,E935	
E942	C9	ret		

E943	CD3FDD	call	DD3F	aller chercher prochain élément de la ligne
E946	C8	ret	z	ignorer les espaces
E947	FE0E	cp	0E	
E949	381D	Jr	c,E968	
E94B	FE20	cp	20	'5'
E94D	3829	Jr	c,E978	
E94F	FE22	cp	22	''''
E951	2809	Jr	z,E95C	ignorer chaîne
E953	FE7C	cp	7C	'EB1EA'
E955	2819	Jr	z,E970	
E957	FEFF	cp	FF	fonction ?
E959	C0	ret	nz	
E95A	23	inc	hl	
E95B	C9	ret		
E95C	23	inc	hl	
E95D	7E	ld	a,(hl)	
E95E	FE22	cp	22	''''
E960	C8	ret	z	
E961	B7	or	a	
E962	20F8	Jr	nz,E95C	
E964	2B	dec	hl	
E965	3E22	ld	a,22	''''
E967	C9	ret		
E968	FE08	cp	08	
E96A	C8	ret	z	

BASIC 1.0

E96B	FE07	cp	07	
E96D	C8	ret	z	
E96E	23	inc	hl	
E96F	23	inc	hl	
E970	F5	push	af	
E971	23	inc	hl	
E972	7E	ld	a,(hl)	
E973	17	rla		
E974	30FB	jr	nc,E971	
E976	F1	pop	af	
E977	C9	ret		
E978	FE18	cp	18	constante de chiffres ?
E97A	D8	ret	c	
E97B	FE19	cp	19	nombre sur un octet ?
E97D	2808	jr	z,E987	
E97F	FE1F	cp	1F	nombre à virgule flottante ?
E981	3803	jr	c,E986	non, nombre sur deux octets
E983	23	inc	hl	
E984	23	inc	hl	ignorer nombre correspondant
E985	23	inc	hl	d'octets
E986	23	inc	hl	
E987	23	inc	hl	
E988	C9	ret		
E989	C5	push	bc	
E98A	D5	push	de	
E98B	E5	push	hl	
E98C	0196E9	ld	bc,E996	
E98F	CDFFE8	call	E8FF	
E992	E1	pop	hl	
E993	D1	pop	de	
E994	C1	pop	bc	
E995	C9	ret		

E996	E5	push	hl	
E997	CD43E9	call	E943	aller chercher prochain élément de la ligne
E99A	D1	pop	de	

BASIC 1.0

E99B	FE02	cp	02	fin de la ligne ?
E99D	D8	ret	c	oui
E99E	FE0E	cp	0E	
E9A0	30F4	Jr	nc,E996	
E9A2	FE07	cp	07	
E9A4	28F0	Jr	z,E996	
E9A6	FE08	cp	08	
E9A8	28EC	Jr	z,E996	
E9AA	EB	ex	de,h1	
E9AB	CD3FDD	call	DD3F	ignorer les espaces
E9AE	FE0D	cp	0D	
E9B0	3802	Jr	c,E9B4	
E9B2	360D	ld	(h1),0D	
E9B4	23	inc	h1	
E9B5	3600	ld	(h1),00	
E9B7	23	inc	h1	
E9B8	3600	ld	(h1),00	
E9BA	EB	ex	de,h1	
E9BB	18D9	Jr	E996	
*****				instruction Basic RUN
E9BD	CD51DD	call	DD51	fin de l'instruction ?
E9C0	EB	ex	de,h1	
E9C1	2A81AE	ld	h1,(AE81)	début de programme comme défaut
E9C4	EB	ex	de,h1	
E9C5	381C	Jr	c,E9E3	oui, fin de l'instruction
E9C7	FE1E	cp	1E	numéro de ligne ?
E9C9	2815	Jr	z,E9E0	
E9CB	FE1D	cp	1D	adresse de ligne ?
E9CD	2811	Jr	z,E9E0	
E9CF	CD0DEA	call	EA0D	
E9D2	2130EA	ld	h1,EA30	charger programme à partir de K7
E9D5	D213BD	Jp	nc,BD13	MC BOOT PROGRAM
E9D8	CDA8EB	call	EBA8	tester type de fichier
E9DB	2A81AE	ld	h1,(AE81)	début du programme
E9DE	1811	Jr	E9F1	
E9E0	CD67E7	call	E767	aller chercher adresse de ligne
E9E3	D5	push	de	
E9E4	CDADD2	call	D2AD	interrompre I/O cassette

BASIC 1.0

E9E7	CD8CC1	call	C18C	
E9EA	CD7AC1	call	C17A	CLEAR
E9ED	CD5EC1	call	C15E	
E9F0	E1	pop	h1	
E9F1	23	inc	h1	
E9F2	F1	pop	af	
E9F3	C393DD	jp	DD93	à la boucle de l'interpréteur
*****				instruction Basic LOAD
E9F6	CD0DEA	call	EA0D	
E9F9	3006	jr	nc,EA01	
E9FB	CDA8EB	call	EBA8	
E9FE	C364C0	jp	C064	au mode READY
EA01	E5	push	h1	
EA02	CD01F5	call	F501	tester si place en mémoire
EA05	CD30EA	call	EA30	charger programme
EA08	CA6BCB	jp	z,CB6B	
EA0B	E1	pop	h1	
EA0C	C9	ret		
EA0D	CD8FEB	call	EB8F	aller chercher nom, ouvrir fichier
EA10	E60E	and	0E	type de fichier
EA12	EE02	xor	02	
EA14	280B	jr	z,EA21	
EA16	CD4ADD	call	DD4A	fin de l'instruction, sinon 'Syntax error'
EA19	CD8CC1	call	C18C	
EA1C	CD6BC1	call	C16B	
EA1F	37	scf		
EA20	C9	ret		
EA21	CD55DD	call	DD55	virgule suit ?
EA24	DC91CE	call	c,CE91	oui, aller chercher valeur 16 bits
EA27	ED533FAE	ld	(AE3F),de	adresse de début
EA2B	CD4ADD	call	DD4A	fin de l'instruction, sinon 'Syntax error'
EA2E	B7	or	a	
EA2F	C9	ret		

BASIC 1.0

EA30	2A3FAE	ld	h1,(AE3F)	adresse de début
EA33	CD83BC	call	BC83	CAS IN DIRECT
EA36	E5	push	h1	
EA37	DC7ABC	call	c,BC7A	CAS IN CLOSE
EA3A	E1	pop	h1	
EA3B	C9	ret		

				instruction Basic CHAIN
EA3C	EEAB	xor	AB	'MERGE'
EA3E	2004	Jr	nz,EA44	
EA40	CD3FDD	call	DD3F	ignorer les espaces
EA43	37	scf		
EA44	9F	sbc	a,a	
EA45	3241AE	ld	(AE41),a	flag pour MERGE
EA48	CD8FEB	call	EB8F	aller chercher nom, ouvrir fichier
EA4B	110000	ld	de,0000	valeur défaut zéro pour ligne début
EA4E	CD55DD	call	DD55	virgule suit ?
EA51	3006	Jr	nc,EA59	non
EA53	7E	ld	a,(h1)	
EA54	FE2C	cp	2C	','
EA56	C491CE	call	nz,CE91	aller chercher valeur 16 bits
EA59	D5	push	de	ranger comme ligne de début
EA5A	CD55DD	call	DD55	virgule suit ?
EA5D	3E00	ld	a,00	
EA5F	3009	Jr	nc,EA6A	non
EA61	CD37DD	call	DD37	tester si encore un caractère
EA64	92	db	92	'DELETE'
EA65	CD37E7	call	E737	vider zone de lignes
EA68	3EFF	ld	a,FF	mettre flag pour DELETE
EA6A	F5	push	af	
EA6B	CD4ADD	call	DD4A	fin de l'instruction, sinon 'Syntax error'
EA6E	CD1BFB	call	FB1B	
EA71	CD3EFC	call	FC3E	Garbage Collection
EA74	CD89E9	call	E989	
EA77	CDD2D5	call	D5D2	supprimer fonctions FN
EA7A	CD49F5	call	F549	
EA7D	F1	pop	af	
EA7E	C5	push	bc	
EA7F	D5	push	de	

BASIC 1.0

EA80	B7	or	a	DELETE ?
EA81	C45AE7	call	nz,E75A	oui, supprimer lignes
EA84	3A41AE	ld	a,(AE41)	flag pour MERGE
EA87	B7	or	a	
EA88	2008	jr	nz,EA92	oui, CHAIN MERGE
EA8A	CD6BC1	call	C16B	supprimer variables
EA8D	CDA8EB	call	EBA8	examiner type de fichier
EA90	1803	Jr	EA95	

EA92	CD9DEB	call	EB9D	tester type de fichier
EA95	D1	pop	de	longueur des variables
EA96	C1	pop	bc	longueur de la zone des chaînes
EA97	CD71F5	call	F571	décaler les chaînes
EA9A	D1	pop	de	aller chercher ligne de début
EA9B	2A81AE	ld	hl,(AE81)	début du programme comme défaut
EA9E	7A	ld	a,d	
EA9F	B3	or	e	pas ligne de début
EAA0	C8	ret	z	
EAA1	CD9AE7	call	E79A	chercher ligne Basic
EAA4	2B	dec	hl	
EAA5	C9	ret		

EAA6	CD8FEB	call	EB8F	instruction Basic MERGE
EAA9	CD4ADD	call	DD4A	aller chercher nom, ouvrir fichier fin de l'instruction, sinon 'Syntax error'
EAAC	CD8CC1	call	C18C	supprimer les variables
EAAF	CD9DEB	call	EB9D	tester type de fichier
EAB2	C364C0	Jp	C064	au mode READY

EAB5	CD7AC1	call	C17A	
EAB8	CD87E6	call	E687	
EABB	2A83AE	ld	hl,(AE83)	fin du programme
EABE	EB	ex	de,hl	
EABF	2A81AE	ld	hl,(AE81)	début du programme
EAC2	23	inc	hl	
EAC3	2283AE	ld	(AE83),hl	fin du programme
EAC6	EB	ex	de,hl	
EAC7	CDDAFF	call	FFDA	bc := hl - de

BASIC 1.0

EACA	EB	ex	de,hl	
EACB	2A8DB0	ld	hl,(B08D)	début des chaînes
EACE	EB	ex	de,hl	
EACF	2B	dec	hl	
EAD0	CDF5FF	call	FFF5	laddr
EAD3	13	inc	de	
EAD4	EB	ex	de,hl	
EAD5	E5	push	hl	
EAD6	2A83AE	ld	nl,(AE83)	fin du programme
EAD9	112000	ld	de,0020	
EADC	19	add	hl,de	
EADD	EB	ex	de,hl	
EADE	E1	pop	hl	
EADF	CDB8FF	call	FFB8	comparer hl <> de
EAE2	3850	jr	c,EB34	'Memory full'
EAE4	CD84EB	call	EB84	
EAE7	B3	or	e	
EAE8	2830	jr	z,EB1A	
EAEA	D5	push	de	
EAEB	CD84EB	call	.EB84	
EAE E	E5	push	hl	
EAEF	7E	ld	a,(hl)	
EAF0	23	inc	hl	
EAF1	B6	or	(hl)	
EAF2	2812	jr	z,EB06	
EAF4	23	inc	hl	
EAF5	7E	ld	a,(hl)	
EAF6	23	inc	hl	
EAF7	66	ld	h,(hl)	
EAF8	6F	ld	l,a	
EAF9	CDB8FF	call	FFB8	comparer hl <> de
EAF C	E1	pop	hl	
EAFD	280F	jr	z,EBOE	
EAFF	3006	jr	nc,EB07	
EB01	CD48EB	call	EB48	
EB04	18E8	jr	EAE E	
EB06	E1	pop	hl	
EB07	E3	ex	(sp),hl	
EB08	CD5EEB	call	EB5E	

BASIC 1.0

EB0B	E1	pop	hl	
EB0C	18C7	Jr	EAD5	
EB0E	E3	ex	(sp),hl	
EB0F	CD5EEB	call	EB5E	
EB12	E1	pop	hl	
EB13	5E	ld	e,(hl)	
EB14	23	inc	hl	
EB15	56	ld	d,(hl)	
EB16	2B	dec	hl	
EB17	19	add	hl,de	
EB18	18BB	Jr	EAD5	
EB1A	7E	ld	a,(hl)	
EB1B	23	inc	hl	
EB1C	B6	or	(hl)	
EB1D	2B	dec	hl	
EB1E	2805	Jr	z,EB25	
EB20	CD48EB	call	EB48	
EB23	18F5	Jr	EB1A	
EB25	2A83AE	ld	hl,(AE83)	fin du programme
EB28	3600	ld	(hl),00	
EB2A	23	inc	hl	
EB2B	3600	ld	(hl),00	
EB2D	23	inc	hl	
EB2E	2283AE	ld	(AE83),hl	fin du programme
EB31	C3B1D5	Jp	D5B1	
EB34	1E07	ld	e,07	'Memory full'
EB36	1802	Jr	EB3A	
EB38	1E18	ld	e,18	'EOF met'
EB3A	D5	push	de	
EB3B	CDADD2	call	D2AD	interromptre I/O cassette
EB3E	CD8CC1	call	C18C	
EB41	CD6BC1	call	C16B	
EB44	D1	pop	de	
EB45	C394CA	Jp	CA94	sortir message d'erreur

BASIC 1.0

EB48	C5	push	bc	
EB49	D5	push	de	
EB4A	E5	push	hl	
EB4B	4E	ld	c,(hl)	
EB4C	23	inc	hl	
EB4D	46	ld	b,(hl)	
EB4E	2A83AE	ld	hl,(AE83)	fin du programme
EB51	EB	ex	de,hl	
EB52	E1	pop	hl	
EB53	CDF2FF	call	FFF2	ldir
EB56	EB	ex	de,hl	
EB57	2283AE	ld	(AE83),hl	fin du programme
EB5A	EB	ex	de,hl	
EB5B	D1	pop	de	
EB5C	C1	pop	bc	
EB5D	C9	ret		
EB5E	D5	push	de	
EB5F	EB	ex	de,hl	
EB60	2A83AE	ld	hl,(AE83)	fin du programme
EB63	73	ld	(hl),e	
EB64	23	inc	hl	
EB65	72	ld	(hl),d	
EB66	23	inc	hl	
EB67	EB	ex	de,hl	
EB68	E3	ex	(sp),hl	
EB69	EB	ex	de,hl	
EB6A	73	ld	(hl),e	
EB6B	23	inc	hl	
EB6C	72	ld	(hl),d	
EB6D	23	inc	hl	
EB6E	D1	pop	de	
EB6F	1B	dec	de	
EB70	1B	dec	de	
EB71	1B	dec	de	
EB72	1B	dec	de	
EB73	7A	ld	a,d	
EB74	B3	or	e	
EB75	2809	Jr	z,EB80	
EB77	CD80BC	call	BC80	CAS IN CHAR

BASIC 1.0

EB7A	30BC	jr	nc,EB38	'EOF met'
EB7C	77	ld	(h1),a	
EB7D	23	inc	h1	
EB7E	18F2	jr	EB72	
EB80	2283AE	ld	(AE83),h1	fin du programme
EB83	C9	ret		
EB84	CD80BC	call	BC80	CAS IN CHAR
EB87	5F	ld	e,a	
EB88	DC80BC	call	c,BC80	CAS IN CHAR
EB8B	30AB	jr -	nc,EB38	'EOF met'
EB8D	57	ld	d,a	
EB8E	C9	ret		
EB8F	CDADD2	call	D2AD	interrompre I/O cassette
EB92	CD6AD2	call	D26A	aller chercher nom, ouvrir fichier
EB95	3242AE	ld	(AE42),a	ranger type de fichier
EB98	ED4343AE	ld	(AE43),bc	ranger longueur de fichier
EB9C	C9	ret		
EB9D	3A42AE	ld	a,(AE42)	type de fichier
EBA0	B7	or	a	
EBA1	CAB5EA	jp	z,EAB5	
EBA4	FE16	cp	16	fichier ASCII ?
EBA6	200B	jr	nz,EBB3	'File type error'
EBA8	3A42AE	ld	a,(AE42)	type de fichier
EBAB	FE16	cp	16	fichier ASCII ?
EBAD	2840	jr	z,EBEF	
EBAF	E6FE	and	FE	annuler bit 0 (fichier protégé)
EBB1	2805	jr	z,EBB8	
EBB3	1E19	ld	e,19	'File type error'
EBB5	C394CA	jp	CA94	sortir message d'erreur
EBB8	CD7AC1	call	C17A	
EBBB	2A81AE	ld	h1,(AE81)	début du programme
EBBE	23	inc	h1	
EBBF	EB	ex	de,h1	
EBC0	2A8DB0	ld	h1,(B08D)	début des chaînes
EBC3	0180FF	ld	bc,FF80	

BASIC 1.0

EBC6	09	add	hl,bc	
EBC7	ED4B43AE	ld	bc,(AE43)	longueur du fichier
EBCB	CDCFFF	call	FFCF	hl := hl - de
EBCE	D4BEFF	call	nc,FFBE	comparer hl <> bc
EBD1	DA34EB	jp	c,EB34	'Memory full'
EBD4	60	ld	h,b	
EBD5	69	ld	l,c	
EBD6	19	add	hl,de	
EBD7	2283AE	ld	(AE83),hl	fin du programme
EBDA	3A42AE	ld	a,(AE42)	type de fichier
EBDD	1F	rra		protégé ?
EBDE	9F	sbc	a,a	
EBDF	3245AE	ld	(AE45),a	fixer flag pour programme protégé
EBE2	EB	ex	de,hl	
EBE3	CD83BC	call	BC83	CAS IN DIRECT
EBE6	CA38EB	jp	z,EB38	'EOF met'
EBE9	CDB1D5	call	D5B1	
EBEC	C398D2	jp	D298	CLOSEIN
EBEF	CD7AC1	call	C17A	
EBF2	CDCBDD	call	DDCB	adresse de ligne actuelle sur zéro
EBF5	CD4CCA	call	CA4C	ligne de cassette dans buffer d'entrée
EBF8	D298D2	jp	nc,D298	CLOSEIN
EBFB	CDBCE6	call	E6BC	convertir ligne en code interpréteur
EBFE	38F5	jr	c,EBF5	pas instruction directe ?
EC00	1E15	ld	e,15	'Direct command found'
EC02	2802	jr	z,EC06	
EC04	1E06	ld	e,06	'Overflow'
EC06	C394CA	jp	CA94	sortir message d'erreur
*****				instruction Basic SAVE
EC09	CDADD2	call	D2AD	interromptre I/O cassette
EC0C	CD56D2	call	D256	OPENOUT
EC0F	0600	ld	b,00	type de fichier 0, programme Basic
EC11	CD55DD	call	DD55	tester si virgule
EC14	3029	jr	nc,EC3F	
EC16	CD37DD	call	DD37	tester si encore un caractère
EC19	0D	db	0D	'variable numérique' (A,B,P)
EC1A	23	inc	hl	

BASIC 1.0

```

EC1B 23      inc    hl
EC1C 7E      ld     a,(hl)    nom de variable
EC1D 23      inc    hl
EC1E E6DF    and    DF            convertir minuscules en majuscules
EC20 F238EC  jp     p,EC38        'Syntax error'
EC23 E5      push   hl
EC24 212CEC  ld     hl,EC2C        adresse de base de la table
EC27 CD93FF  call  FF93            parcourir la table
EC2A E3      ex     (sp),hl
EC2B C9      ret

```

```

EC2C 03      db     03            nombre d'entrées dans la table
EC2D 38EC    dw    EC38            adresse de retour si pas trouvé
EC2F C1      db     C1            'A'
EC30 87EC    dw    EC87            '
EC32 C2      db     C2            'B'
EC33 5CEC    dw    ECE5            '
EC35 D0      db     D0            'P'
EC36 3DEC    dw    EC3D

```

```

EC38 1E02    ld     e,02            'Syntax error'
EC3A C394CA  jp    CA94            sortir message d'erreur

```

```

EC3D 0601    ld     b,01            SAVE ,P
EC3F CD4ADD  call  DD4A            type de fichier 1, protected
                                     fin de l'instruction, sinon 'Syntax
                                     error'

EC42 E5      push   hl
EC43 C5      push   bc
EC44 CD87E6  call  E687
EC47 CD89E9  call  E989
EC4A 2A81AE  ld     hl,(AE81)      début du programme
EC4D 23      inc    hl
EC4E EB      ex     de,hl
EC4F 2A83AE  ld     hl,(AE83)      fin du programme
EC52 C8CFFF  call  FFCF            hl := hl - de
EC55 EB      ex     de,hl
EC56 F1      pop    af

```

BASIC 1.0

EC57 010000 ld bc,0000
 EC5A 1823 Jr EC7F

SAVE ,B
 EC5C 0602 ld b,02 type de fichier 2, binaire
 EC5E CD37DD call DD37 tester si encore un caractère
 EC61 2C db 2C ', '
 EC62 CD91CE call CE91 aller chercher valeur 16 bits,
 adresse de début
 EC65 D5 push de
 EC66 CD37DD call DD37 tester si encore un caractère
 EC69 2C db 2C ', '
 EC6A CD91CE call CE91 aller chercher valeur 16 bits,
 adresse de fin
 EC6D D5 push de
 EC6E CD55DD call DD55 virgule suit ?
 EC71 110000 ld de,0000 défaut zéro
 EC74 DC91CE call c,CE91 oui, aller chercher valeur 16 bits,
 adresse d'entrée
 EC77 D5 push de
 EC78 CD4ADD call DD4A fin de l'instruction, sinon 'Syntax
 error'
 EC7B 78 ld a,b type de fichier
 EC7C C1 pop bc adresse d'entrée
 EC7D D1 pop de adresse de fin
 EC7E E3 ex (sp),hl adresse de début
 EC7F CD98BC call BC98 CAS OUT DIRECT
 EC82 D26BCB Jp nc,CB6B interruption par 'ESC' ?
 EC85 1817 Jr EC9E CLOSEOUT

SAVE ,A
 EC87 CD4ADD call DD4A fin de l'instruction, sinon 'Syntax
 error'
 EC8A E5 push hl
 EC8B 3E09 ld a,09 9
 EC8D CDA2C1 call C1A2 sortie sur canal 9, cassette
 EC90 F5 push af
 EC91 010100 ld bc,0001 1
 EC94 11FFFF ld de,FFFF à 65535
 EC97 CD0DE1 call E10D lister lignes

BASIC 1.0

EC9A	F1	pop	af	
EC9B	CDA2C1	call	C1A2	sortie à nouveau sur défaut
EC9E	CDA1D2	call	D2A1	CLOSEOUT
ECA1	E1	pop	hl	
ECA2	C9	ret		

ECA3	CD44ED	call	ED44	
ECA6	2005	jr	nz,ECAD	
ECA8	CD61DD	call	DD61	ignorer espace, TAB et LF
ECAB	182F	jr	ECDC	

ECAD	FE26	cp	26	'&'
ECAF	281C	jr	z,ECCD	
ECB1	CD7FFF	call	FF7F	tester si numérique
ECB4	3826	jr	c,ECDC	
ECB6	CD10FF	call	FF10	type sur entier
ECB9	CDF3FE	call	FEF3	supprimer variable
ECBC	37	scf		
ECBD	C9	ret		

ECBE	E5	push	hl	
ECBF	CDC6EC	call	ECC6	
ECC2	D1	pop	de	
ECC3	D8	ret	c	
ECC4	EB	ex	de,hl	
ECC5	C9	ret		

ECC6	1600	ld	d,00	
ECC8	7E	ld	a,(hl)	
ECC9	FE26	cp	26	'&'
ECCB	200F	jr	nz,ECDC	
ECCD	CD1CEE	call	EE1C	
ECD0	EB	ex	de,hl	
ECD1	F5	push	af	
ECD2	CD0DFF	call	FF0D	accepter nombre entier hl
ECD5	F1	pop	af	
ECD6	EB	ex	de,hl	
ECD7	D8	ret	c	
ECD8	C8	ret	z	

BASIC 1.0

ECD9	C3F3CA	Jp	CAF3	
ECDC	E5	push	h1	
ECDD	7E	ld	a,(h1)	
ECDE	23	inc	h1	
ECDF	FE2E	cp	2E	','
ECE1	CC61DD	call	z,DD61	ignorer espace, TAB et LF
ECE4	CD83FF	call	FF83	tester si chiffre
ECE7	E1	pop	h1	
ECE8	3806	jr	c,ECF0	
ECEA	7E	ld	a,(h1)	
ECEB	EE2E	xor	2E	','
ECED	C0	ret	nz	
ECEE	23	inc	h1	
ECEF	C9	ret		
ECF0	CD10FF	call	FF10	type sur entier
ECF3	D5	push	de	
ECF4	010000	ld	bc,0000	
ECF7	1146AE	ld	de,AE46	
ECFA	CD53ED	call	ED53	
ECFD	FE2E	cp	2E	','
ECFF	200B	jr	nz,ED0C	
ED01	CDC9ED	call	EDC9	
ED04	CD19FF	call	FF19	type sur 'Real'
ED07	0C	inc	c	
ED08	CD53ED	call	ED53	
ED0B	0D	dec	c	
ED0C	F5	push	af	
ED0D	3EFF	ld	a,FF	
ED0F	12	ld	(de),a	
ED10	F1	pop	af	
ED11	CD77ED	call	ED77	
ED14	D1	pop	de	
ED15	5F	ld	e,a	
ED16	E5	push	h1	
ED17	D5	push	de	
ED18	2146AE	ld	h1,AE46	
ED1B	CDCEED	call	EDCE	
ED1E	D1	pop	de	

BASIC 1.0

ED1F	CD27FF	call	FF27	tester si chaîne
ED22	3008	jr	nc,ED2C	
ED24	E5	push	hl	
ED25	42	ld	b,d	
ED26	CD06FE	call	FE06	
ED29	E1	pop	hl	
ED2A	3811	jr	c,ED3D	
ED2C	7A	ld	a,d	
ED2D	4E	ld	c,(hl)	
ED2E	23	inc	hl	
ED2F	CD94BD	call	BD94	d'entier 4 octets*256 en nombre à virgule flottante
ED32	7B	ld	a,e	
ED33	CD55BD	call	BD55	multiplier nombre par 10^a
ED36	EB	ex	de,hl	
ED37	CD16FF	call	FF16	fixer type de variable sur virgule flottante
ED3A	DC3DBD	call	c,BD3D	variable de (de) à (hl)
ED3D	3E0A	ld	a,0A	
ED3F	E1	pop	hl	
ED40	D8	ret	c	
ED41	C3F3CA	jp	CAF3	
ED44	CD61DD	call	DD61	ignorer espace, TAB et LF
ED47	23	inc	hl	
ED48	16FF	ld	d,FF	
ED4A	FE2D	cp	2D	'-'
ED4C	C8	ret	z	
ED4D	14	inc	d	
ED4E	FE2B	cp	2B	'+'
ED50	C8	ret	z	
ED51	2B	dec	hl	
ED52	C9	ret		
ED53	E5	push	hl	
ED54	CD61DD	call	DD61	ignorer espace, TAB et LF
ED57	23	inc	hl	
ED58	CD83FF	call	FF83	tester si chiffre
ED5B	3804	jr	c,ED61	
ED5D	E1	pop	hl	

BASIC 1.0

ED5E	C38AFF	jp	FF8A	convertir minuscules en majuscules
ED61	E3	ex	(sp),h1	
ED62	E1	pop	h1	
ED63	D630	sub	30	'0'
ED65	12	ld	(de),a	
ED66	B0	or	b	
ED67	2807	jr	z,ED70	
ED69	78	ld	a,b	
ED6A	04	inc	b	
ED6B	FE0C	cp	0C	
ED6D	3001	jr	nc,ED70	
ED6F	13	inc	de	
ED70	79	ld	a,c	
ED71	B7	or	a	
ED72	28DF	jr	z,ED53	
ED74	0C	inc	c	
ED75	18DC	jr	ED53	
ED77	FE45	cp	45	'E'
ED79	2010	jr	nz,ED8B	
ED7B	E5	push	h1	
ED7C	CDC9ED	call	EDC9	
ED7F	CD44ED	call	ED44	
ED82	CC61DD	call	z,DD61	ignorer espace, TAB et LF
ED85	CD83FF	call	FF83	tester si chiffre
ED88	3804	jr	c,ED8E	
ED8A	E1	pop	h1	
ED8B	AF	xor	a	
ED8C	181E	jr	EDAC	
ED8E	E3	ex	(sp),h1	
ED8F	E1	pop	h1	
ED90	CD19FF	call	FF19	fixer type sur 'Real'
ED93	D5	push	de	
ED94	C5	push	bc	
ED95	CD35EE	call	EE35	
ED98	3009	jr	nc,EDA3	
ED9A	7B	ld	a,e	
ED9B	D664	sub	64	100

BASIC 1.0

ED9D	7A	ld	a,d	
ED9E	DE00	sbc	a,00	
EDA0	7B	ld	a,e	
EDA1	3802	jr	c,EDA5	
EDA3	3E7F	ld	a,7F	
EDA5	C1	pop	bc	
EDA6	D1	pop	de	
EDA7	14	inc	d	
EDA8	2002	jr	nz,EDAC	
EDAA	2F	cpl	a	
EDAB	3C	inc	a	
EDAC	C680	add	a,80	
EDAE	5F	ld	e,a	
EDAF	78	ld	a,b	
EDB0	D60C	sub	0C	
EDB2	3001	jr	nc,EDB5	
EDB4	AF	xor	a	
EDB5	91	sub	c	
EDB6	3009	jr	nc,EDC1	
EDB8	83	add	a,e	
EDB9	3801	jr	c,EDBC	
EDBB	AF	xor	a	
EDBC	FE01	cp	01	
EDBE	CE80	adc	a,80	
EDCO	C9	ret		
EDC1	83	add	a,e	
EDC2	3002	jr	nc,EDC6	
EDC4	3EFF	ld	a,FF	
EDC6	D680	sub	80	
EDC8	C9	ret		
EDC9	CD61DD	call	DD61	ignorer espace, TAB et LF
EDCC	23	inc	hl	
EDCD	C9	ret		
EDCE	EB	ex	de,hl	
EDCF	2158AE	ld	hl,AE58	
EDD2	010105	ld	bc,0501	
EDD5	2B	dec	hl	

BASIC 1.0

EDD6	3600	ld	(hl),00	
EDD8	10FB	djnz	EDD5	
EDDA	1A	ld	a,(de)	
EDDB	FEFF	cp	FF	
EDDD	C8	ret	z	
EDDE	77	ld	(hl),a	
EDDF	2153AE	ld	hl,AE53	
EDE2	13	inc	de	
EDE3	1A	ld	a,(de)	
EDE4	FEFF	cp	FF	
EDE6	C8	ret	z	
EDE7	D5	push	de	
EDE8	41	ld	b,c	
EDE9	1600	ld	d,00	
EDEB	E5	push	hl	
EDEC	5E	ld	e,(hl)	
EDED	62	ld	h,d	
EDEE	6B	ld	l,e	
EDEF	29	add	hl,hl	
EDF0	29	add	hl,hl	
EDF1	19	add	hl,de	fois 10
EDF2	29	add	hl,hl	
EDF3	5F	ld	e,a	plus chiffre suivant
EDF4	19	add	hl,de	
EDF5	5D	ld	e,l	
EDF6	7C	ld	a,h	
EDF7	E1	pop	hl	
EDF8	73	ld	(hl),e	
EDF9	23	inc	hl	
EDFA	10EF	djnz	EDEB	
EDFC	D1	pop	de	
EDFD	B7	or	a	
EDFE	28DF	jr	z,EDDF	
EE00	77	ld	(hl),a	
EE01	0C	inc	c	
EE02	18DB	jr	EDDF	
EE04	C5	push	bc	
EE05	E5	push	hl	
EE06	CD35EE	call	EE35	

BASIC 1.0

EE09	EB	ex	de,h1	
EE0A	CD0DFF	call	FF0D	accepter nombre entier h1
EE0D	EB	ex	de,h1	
EE0E	C1	pop	bc	
EE0F	3006	jr	nc,EE17	
EE11	7A	ld	a,d	
EE12	B3	or	e	
EE13	C6FF	add	a,FF	
EE15	3803	jr	c,EE1A	
EE17	50	ld	d,b	
EE18	59	ld	e,c	
EE19	EB	ex	de,h1	
EE1A	C1	pop	bc	
EE1B	C	ret		
EE1C	23	inc	h1	
EE1D	CD61DD	call	DD61	ignorer espace, TAB et LF
EE20	CD8AFF	call	FF8A	convertir minuscules en majuscules
EE23	0602	ld	b,02	base 2, binaire
EE25	FE58	cp	58	'X'
EE27	2806	jr	z,EE2F	
EE29	0610	ld	b,10	base 16, hex
EE2B	FE48	cp	48	'H'
EE2D	2004	jr	nz,EE33	
EE2F	23	inc	h1	
EE30	CD61DD	call	DD61	ignorer espace, TAB et LF
EE33	1802	jr	EE37	
EE35	060A	ld	b,0A	base 10, décimal
EE37	EB	ex	de,h1	
EE38	CD61EE	call	EE61	convertir chiffre hexa en binaire
EE3B	2600	ld	h,00	
EE3D	6F	ld	l,a	
EE3E	301E	jr	nc,EE5E	
EE40	0E00	ld	c,00	
EE42	CD61EE	call	EE61	convertir chiffre hexa en binaire
EE45	3014	jr	nc,EE5B	
EE47	D5	push	de	
EE48	1600	ld	d,00	
EE4A	5F	ld	e,a	

BASIC 1.0

EE4B	D5	push	de	
EE4C	58	ld	e,b	base du système numérique
EE4D	CDBEBD	call	BDBE	multiplication entiers sans signe
EE50	D1	pop	de	
EE51	3803	Jr	c,EE56	
EE53	19	add	hl,de	
EE54	3002	Jr	nc,EE58	
EE56	0EFF	ld	c,FF	
EE58	D1	pop	de	
EE59	18E7	Jr	EE42	

EE5B	79	ld	a,c
EE5C	FE01	cp	01
EE5E	EB	ex	de,hl
EE5F	78	ld	a,b
EE60	C9	ret	

EE61	1A	ld	a,(de)	convertie chiffre hexa en binaire
EE62	13	inc	de	aller chercher caractère
EE63	CD83FF	call	FF83	tester si chiffre
EE66	380A	Jr	c,EE72	oui
EE68	CD8AFF	call	FF8A	convertir minuscules en majuscules
EE6B	FE41	cp	41	'A'
EE6D	3F	ccf		
EE6E	3005	Jr	nc,EE75	inférieur 'A', erreur
EE70	D607	sub	07	'A'-'('9'+1)
EE72	D630	sub	30	'0'
EE74	B8	cp	b	
EE75	D8	ret	c	pas erreur ?
EE76	1B	dec	de	
EE77	AF	xor	a	annuler carry
EE78	C9	ret		

EE79	CD0DFF	call	FF0D	sortir numéro de ligne
EE7C	CD82EE	call	EE82	accepter nombre entier dans hl
EE7F	C341C3	Jp	C341	convertir en représentation ASCII
				sortir chaîne

convertir nombre entier en ASCII

BASIC 1.0

```
EE82 D5      push  de
EE83 C5      push  bc
EE84 CDC3FC  call  FCC3
EE87 AF      xor    a
EE88 CDA7EE  call  EEA7
EE8B 23      inc   hl
EE8C C1      pop   bc
EE8D D1      pop   de
EE8E C9      ret
```

```
EE8F D5      push  de
EE90 C5      push  bc
EE91 AF      xor    a           zéro
EE92 CD9FEE  call  EE9F       convertir nombre en chaîne formatée
EE95 C1      pop   bc
EE96 D1      pop   de
EE97 7E      ld    a,(hl)
EE98 FE20    cp    20        '5'
EE9A C0      ret    nz
EE9B 23      inc   hl
EE9C C9      ret
```

```
EE9D 3E40    ld    a,40
EE9F 226EAE  ld    (AE6E),hl  convertir nombre en chaîne formatée
EEA2 F5      push  af
EEA3 CDB3FC  call  FCB3
EEA6 F1      pop   af
EEA7 C5      push  bc
EEA8 57      ld    d,a
EEA9 D5      push  de
EEAA EB      ex    de,hl
EEAB 2168AE  ld    hl,AE68
EEAE 3600    ld    (hl),00
EEB0 2270AE  ld    (AE70),hl
EEB3 CDB7F0  call  FOB7
EEB6 D1      pop   de
EEB7 CDD4EE  call  EED4
EEBA CD3DF0  call  F03D
```

BASIC 1.0

EEBD	58	ld	e,b	
EEBE	C1	pop	bc	
EEBF	7B	ld	a,e	
EECO	B7	or	a	
EEC1	CC50F0	call	z,F050	
EEC4	CD5FF0	call	F05F	
EEC7	CD69F0	call	F069	
EECA	CD7CF0	call	F07C	
EECD	7A	ld	a,d	
EECE	1F	rfa		
EECF	D0	ret	nc	
EED0	2B	dec	hl	dépassement,
EED1	3625	ld	(hl),25	'%' devant nombre formaté
EED3	C9	ret		
EED4	7A	ld	a,d	
EED5	87	add	a,a	
EED6	3029	jr	nc,EF01	
EED8	FA27EF	jp	m,EF27	
EEDB	7B	ld	a,e	
EEDC	81	add	a,c	
EEDD	D60A	sub	0A	10
EEDF	FA88EF	jp	m,EF88	
EEE2	1601	ld	d,01	
EEE4	41	ld	b,c	
EEE5	79	ld	a,c	
EEE6	B7	or	a	
EEE7	2815	jr	z,EEFE	
EEE9	83	add	a,e	
EEEA	3D	dec	a	
EEEB	5F	ld	e,a	
EEEC	CD0EF0	call	F00E	
EEEF	0601	ld	b,01	
EEF1	79	ld	a,c	
EEF2	FE07	cp	07	
EEF4	3804	jr	c,EEFA	
EEF6	CB72	bit	6,d	
EEF8	2026	jr	nz,EF20	
EEFA	B8	cp	b	
EEFB	C4A0EF	call	nz,EFA0	

BASIC 1.0

EEFE	C362EF	jp	EF62
EF01	7B	ld	a,e
EF02	B7	or	a
EF03	FA0AEF	jp	m,EF0A
EF06	20DC	jr	nz,EEE4
EF08	41	ld	b,c
EF09	C9	ret	
EF0A	43	ld	b,e
EF0B	CDOEFO	call	FOOE
EF0E	78	ld	a,b
EF0F	B7	or	a
EF10	28F6	jr	z,EF08
EF12	93	sub	e
EF13	58	ld	e,b
EF14	47	ld	b,a
EF15	81	add	a,c
EF16	83	add	a,e
EF17	FAE4EE	jp	m,EEE4
EF1A	CDB4EF	call	EFB4
EF1D	C3A0EF	jp	EFA0
EF20	3E06	ld	a,06
EF22	326EAE	ld	(AE6E),a
EF25	1824	jr	EF4B
EF27	0680	ld	b,80
EF29	CD25F0	call	F025
EF2C	3004	jr	nc,EF32
EF2E	CD96F0	call	F096
EF31	AF	xor	a
EF32	47	ld	b,a
EF33	CC36F0	call	z,F036
EF36	200C	jr	nz,EF44
EF38	04	inc	b
EF39	3A6EAE	ld	a,(AE6E)
EF3C	B7	or	a
EF3D	2805	jr	z,EF44
EF3F	05	dec	b

BASIC 1.0

EF40	3C	inc	a	
EF41	326EAE	ld	(AE6E),a	
EF44	79	ld	a,c	
EF45	B7	or	a	
EF46	2804	jr	z,EF4C	
EF48	83	add	a,e	
EF49	90	sub	b	
EF4A	5F	ld	e,a	
EF4B	78	ld	a,b	
EF4C	F5	push	af	
EF4D	47	ld	b,a	
EF4E	CD8BEF	call	EF8B	
EF51	F1	pop	af	
EF52	B8	cp	b	
EF53	280D	jr	z,EF62	
EF55	1C	inc	e	
EF56	23	inc	hl	
EF57	05	dec	b	
EF58	E5	push	hl	
EF59	7E	ld	a,(hl)	
EF5A	FE2E	cp	2E	','
EF5C	2001	jr	nz,EF5F	
EF5E	23	inc	hl	
EF5F	3631	ld	(hl),31	'1'
EF61	E1	pop	hl	
EF62	3E45	ld	a,45	'E'
EF64	CD6FF0	call	F06F	
EF67	7B	ld	a,e	
EF68	87	add	a,a	

BASIC 1.0

EF69	3E2B	ld	a,2B	'+'
EF6B	3005	jr	nc,EF72	
EF6D	AF	xor	a	
EF6E	93	sub	e	
EF6F	5F	ld	e,a	
EF70	3E2D	ld	a,2D	'-'
EF72	CD6FF0	call	F06F	
EF75	7B	ld	a,e	
EF76	0E2F	ld	c,2F	'0'-1
EF78	0C	inc	c	
EF79	D60A	sub	0A	10
EF7B	30FB	jr	nc,EF78	
EF7D	5F	ld	e,a	
EF7E	79	ld	a,c	
EF7F	CD6FF0	call	F06F	
EF82	7B	ld	a,e	
EF83	C63A	add	a,3A	'9'+1
EF85	C36FF0	jp	F06F	
EF88	CDB4EF	call	EFB4	
EF8B	CD36F0	call	F036	
EF8E	80	add	a,b	
EF8F	B9	cp	c	
EF90	3005	jr	nc,EF97	
EF92	CDC8EF	call	EFC8	
EF95	1804	jr	EF9B	
EF97	91	sub	c	
EF98	C4EFEF	call	nz,EFEF	
EF9B	3A6EAE	ld	a,AE6E	
EF9E	B7	or	a	
EF9F	C8	ret	z	
EFA0	0E2E	ld	c,2E	','
EFA2	78	ld	a,b	
EFA3	C5	push	bc	
EFA4	47	ld	b,a	
EFA5	04	inc	b	
EFA6	85	add	a,l	
EFA7	6F	ld	l,a	
EFA8	8C	adc	a,h	

BASIC 1.0

EFA9	95	sub	l	
EFAA	67	ld	h,a	
EFAB	2B	dec	hl	
EFAC	79	ld	a,c	
EFAD	4E	ld	c,(hl)	
EFAE	77	ld	(hl),a	
EFAF	05	dec	b	
EFB0	20F9	jr	nz,EFAB	
EFB2	C1	pop	bc	
EFB3	C9	ret		
EFB4	7B	ld	a,e	
EFB5	81	add	a,c	
EFB6	47	ld	b,a	
EFB7	F0	ret	p	
EFB8	2F	cpl	a	
EFB9	3C	inc	a	
EFBA	0614	ld	b,14	
EFBC	B8	cp	b	
EFBD	3001	jr	nc,EFC0	
EFBF	47	ld	b,a	
EFC0	2B	dec	hl	
EFC1	3630	ld	(hl),30	'0'
EFC3	0C	inc	c	
EFC4	05	dec	b	
EFC5	20F9	jr	nz,EFC0	
EFC7	C9	ret		
EFC8	E5	push	hl	
EFC9	4F	ld	c,a	
EFCA	85	add	a,l	
EFCB	6F	ld	l,a	
EFCC	8C	adc	a,h	
EFCD	95	sub	l	
EFCE	67	ld	h,a	
EFCE	7E	ld	a,(hl)	
EFD0	3600	ld	(hl),00	
EFD2	2270AE	ld	(AE70),hl	
EFD5	FE35	cp	35	'5'
EFD7	D4E1EF	call	nc,EFE1	

BASIC 1.0

EFDA	E1	pop	hl	
EFDB	D8	ret	c	
EFDC	2B	dec	hl	
EFDD	3631	ld	(hl),31	'1'
EFDF	04	inc	b	
EFE0	C9	ret		
EFE1	79	ld	a,c	
EFE2	B7	or	a	
EFE3	C8	ret	z	
EFE4	2B	dec	hl	
EFE5	0D	dec	c	
EFE6	7E	ld	a,(hl)	
EFE7	34	inc	(hl)	
EFE8	FE39	cp	39	'9'
EFEA	D8	ret	c	
EFEB	3630	ld	(hl),30	'0'
EFEF	18F2	jr	EFE1	
EFEF	D5	push	de	
EFF0	C5	push	bc	
EFF1	EB	ex	de,hl	
EFF2	47	ld	b,a	
EFF3	7B	ld	a,e	
EFF4	90	sub	b	
EFF5	6F	ld	l,a	
EFF6	9F	sbc	a,a	
EFF7	82	add	a,d	
EFF8	67	ld	h,a	
EFF9	E5	push	hl	
EFFA	0C	inc	c	
EFFB	1804	jr	F001	
EFFD	1A	ld	a,(de)	
EFFE	13	inc	de	
EFFF	77	ld	(hl),a	
F000	23	inc	hl	
F001	0D	dec	c	
F002	20F9	jr	nz,EFFD	
F004	3630	ld	(hl),30	'0'

BASIC 1.0

F006	23	inc	hl	
F007	05	dec	b	
F008	20FA	jr	nz,F004	
F00A	E1	pop	hl	
F00B	C1	pop	bc	
F00C	D1	pop	de	
F00D	C9	ret		
F00E	E5	push	hl	
F00F	2A70AE	ld	hl,(AE70)	
F012	2B	dec	hl	
F013	7E	ld	a,(hl)	
F014	23	inc	hl	
F015	FE30	cp	30	'0'
F017	2005	jr	nz,F01E	
F019	2B	dec	hl	
F01A	0D	dec	c	
F01B	04	inc	b	
F01C	20F4	jr	nz,F012	
F01E	3600	ld	(hl),00	
F020	2270AE	ld	(AE70),hl	
F023	E1	pop	hl	
F024	C9	ret		
F025	CD9BF0	call	F09B	
F028	9F	sbc	a,a	
F029	3C	inc	a	
F02A	47	ld	b,a	
F02B	7A	ld	a,d	
F02C	E604	and	04	
F02E	2801	jr	z,F031	
F030	04	inc	b	
F031	3A6FAE	ld	a,(AE6F)	
F034	90	sub	b	
F035	C9	ret		
F036	3A6EAE	ld	a,(AE6E)	
F039	B7	or	a	
F03A	C8	ret	z	
F03B	3D	dec	a	

BASIC 1.0

F03C	C9	ret		
F03D	7A	ld	a,d	
F03E	E602	and	02	
F040	C8	ret	z	
F041	78	ld	a,b	
F042	D603	sub	03	
F044	D8	ret	c	
F045	C8	ret	z	
F046	F5	push	af	
F047	0E2C	ld	c,2C	' , '
F049	CDA3EF	call	EFA3	
F04C	04	inc	b	
F04D	F1	pop	af	
F04E	18F2	jr	F042	
F050	7A	ld	a,d	
F051	87	add	a,a	
F052	3007	jr	nc,F05B	
F054	C5	push	bc	
F055	CD25F0	call	F025	
F058	C1	pop	bc	
F059	D8	ret	c	
F05A	C8	ret	z	
F05B	3E30	ld	a,30	'0'
F05D	1806	jr	F065	
F05F	7A	ld	a,d	
F060	E604	and	04	
F062	C8	ret	z	
F063	3E24	ld	a,24	'\$'
F065	1C	inc	e	
F066	2B	dec	h1	
F067	77	ld	(h1),a	
F068	C9	ret		
F069	CD9BF0	call	F09B	
F06C	C8	ret	z	
F06D	30F6	jr	nc,F065	
F06F	E5	push	h1	

BASIC 1.0

F070	2A70AE	ld	hl,(AE70)	
F073	77	ld	(hl),a	
F074	23	inc	hl	
F075	3600	ld	(hl),00	
F077	2270AE	ld	(AE70),hl	
F07A	E1	pop	hl	
F07B	C9	ret		
F07C	7A	ld	a,d	
F07D	B7	or	a	
F07E	F0	ret	p	
F07F	3A6FAE	ld	a,(AE6F)	
F082	93	sub	e	
F083	C8	ret	z	
F084	3810	jr	c,F096	
F086	47	ld	b,a	
F087	7A	ld	a,d	
F088	E620	and	20	
F08A	3E2A	ld	a,2A	'*'
F08C	2002	jr	nz,F090	
F08E	3E20	ld	a,20	'5'
F090	2B	dec	hl	
F091	77	ld	(hl),a	
F092	05	dec	b	
F093	20FB	jr	nz,F090	
F095	C9	ret		
F096	7A	ld	a,d	
F097	F601	or	01	
F099	57	ld	d,a	
F09A	C9	ret		
F09B	78	ld	a,b	
F09C	062D	ld	b,2D	'-'
F09E	87	add	a,a	
F09F	380F	jr	c,F0B0	
FOA1	7A	ld	a,d	
FOA2	E698	and	98	
FOA4	EE80	xor	80	
FOA6	37	scf		

BASIC 1.0

FOA7	C8	ret	z	
FOA8	062B	ld	b,2B	'+'
FOAA	E608	and	08	
FOAC	2002	jr	nz,FOB0	
FOAE	0620	ld	b,20	'5'
FOB0	7A	ld	a,d	
FOB1	F6EF	or	EF	
FOB3	C610	add	a,10	
FOB5	78	ld	a,b	
FOB6	C9	ret		
FOB7	E5	push	h1	
FOB8	EB	ex	de,h1	
FOB9	CDDDF0	call	FODD	
FOBC	E1	pop	h1	
FOBD	78	ld	a,b	
FOBE	87	add	a,a	
FOBF	4F	ld	c,a	
FOC0	C8	ret	z	
FOC1	1A	ld	a,(de)	charger octet
FOC2	E60F	and	0F	isoler quartet inférieur
FOC4	C630	add	a,30	'0', en ASCII
FOC6	2B	dec	h1	
FOC7	77	ld	(h1),a	dans buffer
FOC8	1A	ld	a,(de)	charger octet
FOC9	E6F0	and	F0	isoler quartet supérieur
FOCB	1F	rra		
FOCC	1F	rra		décaler vers le bas
FOCD	1F	rra		
FOCE	1F	rra		
FOCF	C630	add	a,30	'0', en ASCII
FOD1	2B	dec	h1	devant
FOD2	77	ld	(h1),a	dans buffer
FOD3	13	inc	de	
FOD4	05	dec	b	
FOD5	20EA	Jr	nz,FOC1	
FOD7	FE30	cp	30	'0'
FOD9	C0	ret	nz	
FODA	0D	dec	c	
FODB	23	inc	h1	

BASIC 1.0

F0DC	C9	ret	
F0DD	1146AE	ld	de,AE46
F0E0	AF	xor	a
F0E1	47	ld	b,a
F0E2	B6	or	(h1)
F0E3	2B	dec	h1
F0E4	2004	jr	nz,FOEA
F0E6	0D	dec	c
F0E7	20F9	jr	nz,FOE2
F0E9	C9	ret	
FOEA	37	scf	
FOEB	8F	adc	a,a
FOEC	30FD	jr	nc,FOEB
FOEE	EB	ex	de,h1
FOEF	D5	push	de
FOF0	57	ld	d,a
FOF1	1811	jr	F104
FOF3	1A	ld	a,(de)
FOF4	1B	dec	de
FOF5	D5	push	de
FOF6	37	scf	
FOF7	8F	adc	a,a
FOF8	57	ld	d,a
FOF9	58	ld	e,b
FOFA	7E	ld	a,(h1)
FOFB	8F	adc	a,a
FOFC	27	daa	
FOFD	77	ld	(h1),a
FOFE	23	inc	h1
FOFF	1D	dec	e
F100	20F8	jr	nz,FOFA
F102	3003	jr	nc,F107
F104	04	inc	b
F105	3601	ld	(h1),01
F107	2146AE	ld	h1,AE46
F10A	7A	ld	a,d
F10B	87	add	a,a

BASIC 1.0

```

F10C 20EA    jr    nz,F0F8
F10E D1      pop   de
F10F 0D      dec   c
F110 20E1    jr    nz,F0F3
F112 EB      ex    de,hl
F113 C9      ret

```

***** conversion en binaire

```

F114 110101  ld    de,0101
F117 1803    jr    F11C

```

***** conversion en hexa

```

F119 110F04  ld    de,040F
F11C D5      push  de
F11D 79      ld    a,c
F11E CD4BFF  call  FF4B    fixer type de variable
F121 E3      ex    (sp),hl
F122 E5      push  hl
F123 C5      push  bc
F124 CDC2FE  call  FEC2    UNT
F127 1157AE  ld    de,AE57
F12A AF      xor   a
F12B 12      ld    (de),a
F12C F1      pop   af
F12D C1      pop   bc
F12E D601    sub   01
F130 CE00    adc   a,00
F132 F5      push  af
F133 7D      ld    a,l
F134 A1      and   c
F135 F6F0    or    F0
F137 27      daa
F138 C6A0    add   a,A0
F13A CE40    adc   a,40
F13C 1B      dec   de
F13D 12      ld    (de),a
F13E 7D      ld    a,l
F13F B1      or    c
F140 A9      xor   c
F141 6F      ld    l,a

```


BASIC 1.0

```

F142 B4      or      h
F143 280E    jr      z,F153
F145 C5      push    bc
F146 7C      ld      a,h
F147 1F      rra
F148 67      ld      h,a
F149 7D      ld      a,l
F14A 1F      rra
F14B 6F      ld      l,a
F14C 05      dec     b
F14D 20F7    jr      nz,F146
F14F C1      pop     bc
F150 F1      pop     af
F151 18DB    jr      F12E

```

```

F153 F1      pop     af
F154 20D8    jr      nz,F12E
F156 E1      pop     hl
F157 C9      ret

```

```

F158 CDC2FE  call   FEC2
F15B E7      rst    4
F15C C30AFF  jp     FFOA

```

fonction Basic PEEK
 UNT
 READ RAM; ld a,(hl)
 accepter contenu accu comme nombre
 entier

```

F15F CD91CE  call   CE91
F162 D5      push   de
F163 CD37DD  call   DD37
F166 2C      db     2C
F167 CD67CE  call   CE67
F16A D1      pop     de
F16B 12      ld     (de),a
F16C C9      ret

```

instruction Basic POKE
 aller chercher adresse 16 bits
 tester si encore un caractère
 ','
 aller chercher valeur 8 bits
 écrire valeur dans adresse

```

F16D CD8DFE  call   FE8D
F170 44      ld     b,h
F171 4D      ld     c,l

```

fonction Basic INP
 CINT
 adresse port dans bc

BASIC 1.0

F172	ED78	in	a,(c)	lire port
F174	C30AFF	jp	FF0A	accepter contenu accu comme nombre entier

F177	CD94F1	call	F194	instruction Basic OUT aller chercher adresse et valeur
F17A	ED79	out	(c),a	sortir
F17C	C9	ret		

F17D	CD94F1	call	F194	instruction Basic WAIT aller chercher valeurs 16 et 8 bits
F180	57	ld	d,a	valeur 8 bits dans d
F181	1E00	ld	e,00	3ème paramètre zéro
F183	2808	jr	z,F18D	aucune autre valeur ?
F185	CD37DD	call	DD37	tester si encore un caractère
F188	2C	db	2C	','
F189	CD67CE	call	CE67	aller chercher valeur 8 bits
F18C	5F	ld	e,a	et dans e
F18D	ED78	in	a,(c)	lire port
F18F	AB	xor	e	
F190	A2	and	d	lier
F191	28FA	jr	z,F18D	et attendre
F193	C9	ret		

F194	CD91CE	call	CE91	aller chercher valeurs 16 et 8 bits
F197	42	ld	b,d	aller chercher valeur 16 bits
F198	4B	ld	c,e	dans bc
F199	CD37DD	call	DD37	tester si encore un caractère
F19C	2C	db	2C	','
F19D	C367CE	jp	CE67	et aller chercher valeur 8 bits

F1A0	23	inc	hl	extension d'instruction augmenter pointeur de programme
F1A1	7E	ld	a,(hl)	
F1A2	B7	or	a	octet nul suit ?
F1A3	2010	jr	nz,F1B5	non, 'Unknown command'
F1A5	23	inc	hl	
F1A6	E5	push	hl	
F1A7	CDD4BC	call	BCD4	KL FIND COMMAND

BASIC 1.0

F1AA	EB	ex	de,hl	adresse de l'instruction dans de
F1AB	E1	pop	hl	
F1AC	3007	Jr	nc,F1B5	pas trouvé, 'Unknown command'
F1AE	7E	ld	a,(hl)	aller chercher caractère
F1AF	23	inc	hl	ignorer mot instruction
F1B0	17	rla		bit 7 mis ?
F1B1	30FB	Jr	nc,F1AE	non
F1B3	180A	Jr	F1BF	à l'instruction CALL
F1B5	1E1C	ld	e,1C	'Unknown command'
F1B7	C394CA	Jp	CA94	sortir message d'erreur

F1BA	CD91CE	call	CE91	instruction Basic CALL aller chercher adresse
F1BD	0EFF	ld	c,FF	FF = Ram sélectionnée
F1BF	ED5372AE	ld	(AE72),de	adresse dans AE72
F1C3	79	ld	a,c	
F1C4	3274AE	ld	(AE74),a	octet de configuration dans AE74
F1C7	ED7377AE	ld	(AE77),sp	sauver pointeur de pile
F1CB	0620	ld	b,20	maximum 32 paramètres
F1CD	CD55DD	call	DD55	virgule suit ?
F1D0	3006	Jr	nc,F1D8	non
F1D2	CD91CE	call	CE91	aller chercher paramètres
F1D5	D5	push	de	et sur la pile
F1D6	10F5	dJnz	F1CD	aller chercher prochain paramètre
F1D8	CD4ADD	call	DD4A	fin de l'instruction, sinon 'Syntax error'
F1DB	2275AE	ld	(AE75),hl	sauver registre hl
F1DE	3E20	ld	a,20	
F1E0	90	sub	b	nombre de paramètres dans accu
F1E1	DD210000	ld	ix,0000	
F1E5	DD39	add	ix,sp	pointeur de pile dans ix
F1E7	DF	rst	3	exécuter routine
F1E8	72AE	dw	AE72	
F1EA	ED7B77AE	ld	sp,(AE77)	rappeler pointeur de pile
F1EE	2A75AE	ld	hl,(AE75)	restaure registre hl
F1F1	C9	ret		

F1F2	3E0D	ld	a,0D	initialiser les TABs 13
------	------	----	------	----------------------------

BASIC 1.0

F1F4	1803	jr	F1F9	
*****				instruction Basic ZONE
F1F6	CD6DCE	call	CE6D	aller chercher valeur 8 bits non nulle
F1F9	3279AE	ld	(AE79),a	ranger écart du tabulateur
F1FC	C9	ret		
*****				instruction Basic PRINT
F1FD	CDC6C1	call	C1C6	numéro de canal
F200	F5	push	af	
F201	CD08F2	call	F208	sortie PRINT
F204	F1	pop	af	
F205	C3A2C1	jp	C1A2	restaurer numéro de canal
F208	CD51DD	call	DD51	fin de l'instruction ?
F20B	DA4EC3	jp	c,C34E	oui, sortir LF
F20E	FEED	cp	ED	'USING'
F210	CAC4F2	jp	z,F2C4	
F213	EB	ex	de,h1	
F214	2124F2	ld	h1,F224	adresse de base de la table
F217	CD93FF	call	FF93	parcourir la table
F21A	EB	ex	de,h1	
F21B	CDFBFF	call	FFFB	jp (de)
F21E	CD51DD	call	DD51	fin de l'instruction ?
F221	30EB	jr	nc,F20E	non, continuer
F223	C9	ret		

F224	05	db	05	nombre d'entrées dans la table
F225	33F2	dw	F233	adresse de retour si pas trouvé
F227	2C	db	2C	','
F228	5CF2	dw	F25C	
F22A	E5	db	E5	'SPC'
F22B	77F2	dw	F277	
F22D	EA	db	EA	'TAB'
F22E	80F2	dw	F280	
F230	3B	db	3B	','
F231	3FDD	dw	DD3F	ignorer espaces

BASIC 1.0

```

*****
F233 CDFBCE call CEFB PRINT
F236 F5 push af aller chercher expression
F237 E5 push hl
F238 CD45FF call FF45 tester si chaîne
F23B 280C jr z,F249 oui
F23D CD9DEE call EE9D convertir nombre en chaîne ASCII
F240 CDDCF7 call F7DC aller chercher paramètres de chaîne
F243 3620 ld (hl),20 '5', ajouter espaces
F245 2AC2B0 ld hl,(B0C2)
F248 34 inc (hl)
F249 2AC2B0 ld hl,(B0C2)
F24C 7E ld a,(hl)
F24D CDB9C2 call C2B9 sélectionner courant de sortie
F250 D44EC3 call nc,C34E sortir LF
F253 CD28F8 call F828 sortir chaîne
F256 E1 pop hl
F257 F1 pop af
F258 CC4EC3 call z,C34E sortir LF
F25B C9 ret

*****
F25C CD3FDD call DD3F PRINT ,
F25F 3A79AE ld a,(AE79) ignorer espaces
F262 4F ld c,a écart de tabulation
F263 CD90C2 call C290
F266 3D dec a
F267 91 sub c
F268 30FD jr nc,F267
F26A 2F cpl a
F26B 3C inc a
F26C 47 ld b,a
F26D 81 add a,c
F26E CDB9C2 call C2B9 sélectionner courant de sortie
F271 D24EC3 jp nc,C34E sortir LF
F274 78 ld a,b
F275 181E jr F295

*****
F277 CDA0F2 call F2A0 PRINT SPC
aller chercher argument entre

```

				parenthèses
F27A	CDAFF2	call	F2AF	
F27D	7B	ld	a,e	
F27E	1815	jr	F295	
*****				PRINT TAB
F280	CDA0F2	call	F2A0	aller chercher argument entre parenthèses
F283	1B	dec	de	
F284	CDAFF2	call	F2AF	
F287	CD90C2	call	C290	
F28A	2F	cpl	a	
F28B	3C	inc	a	
F28C	1C	inc	e	
F28D	83	add	a,e	
F28E	3805	jr	c,F295	
F290	CD4EC3	call	C34E	sortir LF
F293	1D	dec	e	
F294	7B	ld	a,e	
F295	47	ld	b,a	
F296	04	inc	b	
F297	05	dec	b	
F298	C8	ret	z	
F299	3E20	ld	a,20	'5'
F29B	CD56C3	call	C356	sortir
F29E	18F7	Jr	F297	
*****				aller chercher argument entre parenthèses
F2A0	CD3FDD	call	DD3F	ignorer espaces
F2A3	CD37DD	call	DD37	tester si encore un caractère
F2A6	28	db	28	'('
F2A8	CD86CE	call	CE86	aller chercher valeur entière avec signe
F2A9	CD37DD	call	DD37	tester si encore un caractère
F2AD	29	db	29	')'
F2AE	C9	ret		
F2AF	7A	ld	a,d	
F2B0	17	rfa		
F2B1	3003	Jr	nc,F2B6	

BASIC 1.0

F2B3	110000	ld	de,0000	
F2B6	CD9FC2	call	C29F	
F2B9	D0	ret	nc	
F2BA	E5	push	hl	
F2BB	EB	ex	de,hl	
F2BC	5F	ld	e,a	accu comme diviseur
F2BD	1600	ld	d,00	Hi-Byte zéro
F2BF	CDC1BD	call	BDC1	division entiers sans signe
F2C2	E1	pop	hl	reste en de
F2C3	C9	ret		

PRINT USING

F2C4	CD3FDD	call	DD3F	ignorer espaces
F2C7	CDA5CE	call	CEA5	aller chercher expression chaîne
F2CA	CD37DD	call	DD37	tester si encore un caractère
F2CD	3B	db	3B	','
F2CE	E5	push	hl	
F2CF	2AC2B0	ld	hl,(BOC2)	
F2D2	7E	ld	a,(hl)	
F2D3	B7	or	a	
F2D4	2875	jr	z,F34B	'Improper argument'
F2D6	E3	ex	(sp),hl	
F2D7	CDFBCE	call	CEFB	aller chercher expression
F2DA	AF	xor	a	
F2DB	327AAE	ld	(AE7A),a	
F2DE	D1	pop	de	
F2DF	D5	push	de	
F2E0	EB	ex	de,hl	
F2E1	46	ld	b,(hl)	
F2E2	23	inc	hl	
F2E3	7E	ld	a,(hl)	
F2E4	23	inc	hl	
F2E5	66	ld	h,(hl)	
F2E6	6F	ld	l,a	
F2E7	EB	ex	de,hl	
F2E8	CD24F3	call	F324	
F2EB	305E	jr	nc,F34B	'Improper argument'
F2ED	CD51DD	call	DD51	fin de l'instruction ?
F2F0	381D	jr	c,F30F	oui
F2F2	FE3B	cp	3B	','

BASIC 1.0

F2F4	2804	jr	z,F2FA	
F2F6	FE2C	cp	2C	','
F2F8	204C	Jr	nz,F346	'Syntax error'
F2FA	CD3FDD	call	DD3F	ignorer espaces
F2FD	2810	Jr	z,F30F	fin de ligne ?
F2FF	D5	push	de	
F300	CDFBCE	call	CEFB	aller chercher expression
F303	D1	pop	de	
F304	78	ld	a,b	
F305	B7	or	a	
F306	28D6	Jr	z,F2DE	
F308	CD24F3	call	F324	
F30B	30D1	Jr	- nc,F2DE	
F30D	18DE	Jr	F2ED	
F30F	F5	push	af	
F310	3EFF	ld	a,FF	
F312	327AAE	ld	(AE7A),a	
F315	78	ld	a,b	
F316	B7	or	a	
F317	C424F3	call	nz,F324	
F31A	F1	pop	af	
F31B	DC4EC3	call	c,C34E	sortir LF
F31E	E3	ex	(sp),hl	
F31F	CDE8FB	call	FBE8	
F322	E1	pop	hl	
F323	C9	ret		
F324	E5	push	hl	
F325	1A	ld	a,(de)	
F326	FE5F	cp	5F	
F328	2009	Jr	nz,F333	
F32A	78	ld	a,b	
F32B	FE02	cp	02	
F32D	380C	Jr	c,F33B	
F32F	13	inc	de	
F330	05	dec	b	
F331	1808	Jr	F33B	
F333	CD50F3	call	F350	

BASIC 1.0

F336	D4A3F3	call	nc,F3A3
F339	3809	jr	c,F344

BASIC 1.0

F33B	1A	ld	a,(de)	
F33C	CD56C3	call	C356	sortir
F33F	13	inc	de	
F340	05	dec	b	
F341	20E2	Jr	nz,F325	
F343	B7	or	a	
F344	E1	pop	hl	
F345	C9	ret		
F346	1E02	ld	e,02	'Syntax error'
F348	C394CA	Jp	CA94	sortir message d'erreur
F34B	1E05	ld	e,05	'Improper argument'
F34D	C394CA	Jp	CA94	sortir message d'erreur
F350	1A	ld	a,(de)	
F351	FE21	cp	21	'!'
F353	0E01	ld	c,01	
F355	2821	Jr	z,F378	
F357	FE26	cp	26	'&'
F359	0E00	ld	c,00	
F35B	281B	Jr	z,F378	
F35D	EE5C	xor	5C	'Backslash'
F35F	C0	ret	nz	
F360	C5	push	bc	
F361	D5	push	de	
F362	0E02	ld	c,02	
F364	13	inc	de	
F365	05	dec	b	
F366	280A	Jr	z,F372	
F368	1A	ld	a,(de)	
F369	FE5C	cp	5C	'Backslash'
F36B	2809	Jr	z,F376	
F36D	0C	inc	c	
F36E	FE20	cp	20	'5'
F370	28F2	Jr	z,F364	
F372	D1	pop	de	
F373	C1	pop	bc	
F374	B7	or	a	
F375	C9	ret		

BASIC 1.0

F376	F1	pop	af	
F377	F1	pop	af	
F378	13	inc	de	
F379	05	dec	b	
F37A	C5	push	bc	
F37B	D5	push	de	
F37C	3A7AAE	ld	a,(AE7A)	
F37F	B7	or	a	
F380	201D	jr	nz,F39F	
F382	CD3CFF	call	FF3C	type 'chaîne', sinon 'Type mismatch'
F385	79	ld	a,c	
F386	B7	or	a	
F387	F5	push	af	
F388	41	ld	b,c	
F389	0E00	ld	c,00	
F38B	2AC2B0	ld	hl,(B0C2)	
F38E	EB	ex	de,hl	
F38F	C471F9	call	nz,F971	
F392	CD28F8	call	F828	sortir chaîne
F395	F1	pop	af	
F396	2807	jr	z,F39F	
F398	2AC2B0	ld	hl,(B0C2)	
F39B	96	sub	(hl)	
F39C	CD95F2	call	F295	
F39F	D1	pop	de	
F3A0	C1	pop	bc	
F3A1	37	scf		
F3A2	C9	ret		
F3A3	CDBAF3	call	F3BA	tester si caractère de formatage
F3A6	D0	ret	nc	
F3A7	3A7AAE	ld	a,(AE7A)	
F3AA	B7	or	a	
F3AB	200B	jr	nz,F3B8	
F3AD	C5	push	bc	
F3AE	D5	push	de	
F3AF	79	ld	a,c	
F3B0	CD9FEE	call	EE9F	formater nombre
F3B3	CD41C3	call	C341	sortir chaîne jusqu'à (0)
F3B6	D1	pop	de	

BASIC 1.0

```
F3B7 C1      pop      bc
F3B8 37      scf
F3B9 C9      ret
```

***** tester si caractere de formatage

```
F3BA C5      push     bc
F3BB D5      push     de
F3BC 0E80    ld       c,80
F3BE 2600    ld       h,00
F3C0 1A      ld       a,(de)
F3C1 FE2B    cp       2B      '+'
F3C3 2007    jr       nz,F3CC
F3C5 13      inc      de
F3C6 05      dec      b
F3C7 2823    jr       z,F3EC
F3C9 24      inc      h
F3CA 0E88    ld       c,88
F3CC 1A      ld       a,(de)
F3CD FE2E    cp       2E      '.'
F3CF 281F    jr       z,F3F0
F3D1 FE23    cp       23      '#'
F3D3 2839    jr       z,F40E
F3D5 13      inc      de
F3D6 05      dec      b
F3D7 2813    jr       z,F3EC
F3D9 EB      ex       de,h1
F3DA BE      cp       (h1)
F3DB EB      ex       de,h1
F3DC 200E    jr       nz,F3EC
F3DE 24      inc      h
F3DF 24      inc      h
F3E0 2E04    ld       l,04
F3E2 FE24    cp       24      '$'
F3E4 2823    jr       z,F409
F3E6 2E20    ld       l,20      '5'
F3E8 FE2A    cp       2A      '*'
F3EA 2811    jr       z,F3FD
F3EC D1      pop      de
F3ED C1      pop      bc
F3EE B7      or       a
```

BASIC 1.0

F3EF	C9	ret		
F3F0	13	inc	de	
F3F1	05	dec	b	
F3F2	28F8	jr	z,F3EC	
F3F4	1A	ld	a,(de)	
F3F5	FE23	cp	23	'#'
F3F7	20F3	jr	nz,F3EC	
F3F9	1B	dec	de	
F3FA	04	inc	b	
F3FB	1811	jr	F40E	
F3FD	13	inc	de	
F3FE	05	dec	b	
F3FF	280A	jr	z,F40B	
F401	1A	ld	a,(de)	
F402	FE24	cp	24	'\$'
F404	2005	jr	nz,F40B	
F406	24	inc	h	
F407	2E24	ld	l,24	
F409	13	inc	de	
F40A	05	dec	b	
F40B	79	ld	a,c	
F40C	B5	or	l	
F40D	4F	ld	c,a	
F40E	F1	pop	af	
F40F	F1	pop	af	
F410	CD1BF4	call	F41B	
F413	7C	ld	a,h	
F414	85	add	a,l	
F415	FE15	cp	15	
F417	D24BF3	jp	nc,F34B	'Improper argument'
F41A	C9	ret		
F41B	2E00	ld	l,00	
F41D	04	inc	b	
F41E	05	dec	b	
F41F	C8	ret	z	
F420	1A	ld	a,(de)	
F421	FE2E	cp	2E	','

BASIC 1.0

F423	2814	Jr	z,F439	
F425	FE2C	cp	2C	' , '
F427	280A	Jr	z,F433	
F429	FE23	cp	23	' # '
F42B	2015	Jr	nz,F442	
F42D	24	inc	h	
F42E	13	inc	de	
F42F	05	dec	b	
F430	20EE	Jr	nz,F420	
F432	C9	ret		
F433	79	ld	a,c	
F434	F602	or	02	
F436	4F	ld	c,a	
F437	18F4	Jr	F42D	
F439	2C	inc	l	
F43A	13	inc	de	
F43B	05	dec	b	
F43C	C8	ret	z	
F43D	1A	ld	a,(de)	
F43E	FE23	cp	23	' # '
F440	28F7	Jr	z,F439	
F442	EB	ex	de,h1	
F443	E5	push	h1	
F444	FE5E	cp	5E	' ^ '
F446	2018	Jr	nz,F460	
F448	23	inc	h1	
F449	BE	cp	(h1)	
F44A	2014	Jr	nz,F460	
F44C	23	inc	h1	
F44D	BE	cp	(h1)	
F44E	2010	Jr	nz,F460	
F450	23	inc	h1	
F451	BE	cp	(h1)	
F452	200C	Jr	nz,F460	
F454	23	inc	h1	
F455	78	ld	a,b	
F456	D604	sub	04	
F458	3806	Jr	c,F460	

BASIC 1.0

```

F45A 47      ld      b,a
F45B E3      ex      (sp),hl
F45C 79      ld      a,c
F45D F640    or      40
F45F 4F      ld      c,a
F460 E1      pop     hl
F461 EB      ex      de,hl
F462 78      ld      a,b
F463 B7      or      a
F464 C8      ret     z
F465 79      ld      a,c
F466 E608    and     08
F468 C0      ret     nz
F469 1A      ld      a,(de)
F46A FE2D    cp      2D      '-'
F46C 3E10    ld      a,10
F46E 2806    jr      z,F476
F470 1A      ld      a,(de)
F471 FE2B    cp      2B      '+'
F473 C0      ret     nz
F474 3E18    ld      a,18
F476 B1      or      c
F477 4F      ld      c,a
F478 13      inc     de
F479 05      dec     b
F47A C9      ret

```

```

F47B CDC6C1  call   C1C6      instruction Basic WRITE
F47E F5      push   af        numéro de canal présent ?
F47F CD51DD  call   DD51      fin de l'instruction ?
F482 3839    jr     c,F4BD    oui
F484 CDFBCE  call   CEFB      aller chercher expression
F487 F5      push   af
F488 E5      push   hl
F489 CD45FF  call   FF45      tester si chaîne
F48C 280B    jr     z,F499    oui, sortir avec guillemets
F48E CD8FEE  call   EE8F
F491 CDDCF7  call   F7DC
F494 CD28F8  call   F828      sortir chaîne

```

BASIC 1.0

F497	180D	Jr	F4A6	
F499	3E22	ld	a,22	'''
F49B	CD56C3	call	C356	sortir
F49E	CD28F8	call	F828	sortir chaîne
F4A1	3E22	ld	a,22	'''
F4A3	CD56C3	call	C356	sortir
F4A6	E1	pop	hl	
F4A7	F1	pop	af	
F4A8	2813	Jr	z,F4BD	
F4AA	FE3B	cp	3B	','
F4AC	2805	Jr	z,F4B3	
F4AE	FE2C	cp	2C -	','
F4B0	C246F3	jp	nz,F346	'Syntax error'
F4B3	CD3FDD	call	DD3F	ignorer espaces
F4B6	3E2C	ld	a,2C	','
F4B8	CD56C3	call	C356	sortir
F4BB	18C7	Jr	F484	
F4BD	CD4EC3	call	C34E	sortir LF
F4C0	F1	pop	af	
F4C1	C3A2C1	jp	C1A2	
*****				configurer mémoire
F4C4	0100AC	ld	bc,AC00	place en mémoire de de à hl
F4C7	CDBEFF	call	FFBE	comparer hl <> bc
F4CA	DO	ret	nc	adresse supérieure < AC00 ?
F4CB	227BAE	ld	(AE7B),hl	HIMEM
F4CE	228FB0	ld	(B08F),hl	fin des chaînes
F4D1	227DAE	ld	(AE7D),hl	fin de la Ram libre
F4D4	EB	ex	de,hl	
F4D5	227FAE	ld	(AE7F),hl	début de la Ram libre
F4D8	012F01	ld	bc,012F	plus 303
F4DB	09	add	hl,bc	
F4DC	D8	ret	c	
F4DD	281AE	ld	(AE81),hl	donne début du programme
F4E0	EB	ex	de,hl	
F4E1	23	inc	hl	
F4E2	B7	or	a	
F4E3	ED52	sbc	hl,de	

BASIC 1.0

```
F4E5 D8      ret    c
F4E6 7C      ld     a,h
F4E7 FE04    cp     04
F4E9 D8      ret    c
F4EA AF      xor    a
F4EB 3291B0  ld     (B091),a
F4EE C9      ret
```

```
***** instruction Basic MEMORY
F4EF CD3EFC  call   FC3E   Garbage Collection
F4F2 CD91CE  call   CE91   aller chercher valeur 16 bits
F4F5 E5      push  hl
F4F6 CD50F7  call   F750
F4F9 CD75F6  call   F675
F4FC 227BAE  ld     (AE7B),hl  fixer HIMEM
F4FF E1      pop   hl
F500 C9      ret
```

```
***** place pour programme à charger
F501 D5      push  de
F502 2A7FAE  ld     hl,(AE7F)  début de la Ram libre
F505 EB      ex    de,hl
F506 2A7BAE  ld     hl,(AE7B)  HIMEM
F509 CDCFFF  call  FFCF       hl := hl - de
F50C E3      ex    (sp),hl
F50D CDCFFF  call  FFCF       hl := hl - de
F510 D1      pop   de
F511 13      inc   de
F512 CDB8FF  call  FFB8       comparer hl <> de
F515 3803    jr    c,F51A     'Memory full'
F517 2B      dec   hl
F518 09      add  hl,bc
F519 D0      ret  nc
F51A C33EF7  jp    F73E     'Memory full'
```

```
***** calculer longueur de la zone des chaînes
F51D D5      push  de
F51E E5      push  hl
F51F 2A8DB0  ld     hl,(B08D)  début des chaînes
F522 EB      ex    de,hl
```

BASIC 1.0

```

F523 2A8FB0 ld hl,(B08F) fin des chaines
F526 CDDAFF call FFDA bc := hl - de
F529 E1 pop hl
F52A D1 pop de
F52B C9 ret

```

***** augmenter pointeurs de PRG et de variables de bc

```

F52C 2A83AE ld hl,(AE83) fin du programme
F52F 09 add hl,bc
F530 2283AE ld (AE83),hl fin du programme
F533 2A85AE ld hl,(AE85) début des variables
F536 09 add hl,bc
F537 2285AE ld (AE85),hl début des variables
F53A 2A87AE ld hl,(AE87) début des tableaux
F53D 09 add hl,bc
F53E 2287AE ld (AE87),hl début des tableaux
F541 2A89AE ld hl,(AE89) fin des tableaux
F544 09 add hl,bc
F545 2289AE ld (AE89),hl fin des tableaux
F548 C9 ret

```

```

F549 2A85AE ld hl,(AE85) début des variables
F54C EB ex de,hl
F54D 2A87AE ld hl,(AE87) début des tableaux
F550 CDCFFF call FFCF hl := hl - de
F553 E5 push hl
F554 2A89AE ld hl,(AE89) fin des tableaux
F557 CDDAFF call FFDA bc := hl - de
F55A C5 push bc
F55B 2A8DB0 ld hl,(B08D) début des chaines
F55E EB ex de,hl
F55F 2A89AE ld hl,(AE89) fin des tableaux
F562 2B dec hl
F563 78 ld a,b
F564 B1 or c
F565 C4F5FF call nz,FFF5 lddr
F568 EB ex de,hl
F569 228DB0 ld (B08D),hl début des chaines
F56C C1 pop bc

```

BASIC 1.0

F56D D1 pop de
 F56E C3B1D5 jp D5B1 restaurer le pointeur de variable

F571 2A83AE ld hl,(AE83) fin de programme
 F574 2285AE ld (AE85),hl égale début des variables
 F577 EB ex de,hl
 F578 19 add hl,de plus longueur des variables
 F579 2287AE ld (AE87),hl égale début des tableaux
 F57C 2A8DB0 ld hl,(B08D) début des chaînes
 F57F 23 inc hl
 F580 78 ld a,b zone des chaînes
 F581 B1 or c
 F582 C4F2FF call nz,FFF2 présent, alors ldir
 F585 2B dec hl
 F586 228DB0 ld (B08D),hl début des chaînes
 F589 EB ex de,hl
 F58A 2289AE ld (AE89),hl fin des tableaux
 F58D C9 ret

***** initialiser pile Basic

F58E F5 push af
 F58F E5 push hl
 F590 218BAE ld hl,AE8B
 F593 228BB0 ld (B08B),hl pointeur de pile Basic
 F596 3E01 ld a,01
 F598 CDB0F5 call F5B0 réserver place dans pile Basic
 F59B 3600 ld (hl),00
 F59D E1 pop hl
 F59E F1 pop af
 F59F C9 ret

***** libérer place dans pile Basic

F5A0 2A8BBO ld hl,(B08B) pointeur de pile Basic
 F5A3 2F cpl a
 F5A4 3C inc a ôter contenu accu
 F5A5 C8 ret z
 F5A6 85 add a,l
 F5A7 6F ld l,a
 F5A8 3EFF ld a,FF

BASIC 1.0

F5AA	8C	adc	a,h	
F5AB	67	ld	h,a	
F5AC	228BB0	ld	(B08B),hl	pointeur de pile Basic
F5AF	C9	ret		

				réserver place dans pile Basic
F5B0	2A8BB0	ld	hl,(B08B)	pointeur de pile Basic
F5B3	E5	push	hl	
F5B4	85	add	a,l	
F5B5	6F	ld	l,a	additionner contenu accu
F5B6	8C	adc	a,h	
F5B7	95	sub	l	
F5B8	67	ld	h,a	
F5B9	228BB0	ld	(B08B),hl	pointeur de pile Basic
F5BC	3E78	ld	a,78	
F5BE	85	add	a,l	donne plus &4F78 dépassement ?
F5BF	3E4F	ld	a,4F	
F5C1	8C	adc	a,h	alors pointeur de pile > &B088
F5C2	E1	pop	hl	
F5C3	D0	ret	nc	
F5C4	CD8EF5	call	F58E	initialiser pile Basic
F5C7	C33EF7	jp	F73E	'Memory full'

F5CA	2A8FB0	ld	hl,(B08F)	fin des chaînes
F5CD	228DB0	ld	(B08D),hl	début des chaînes
F5D0	C9	ret		

				réserver place pour chaîne
F5D1	2F	cpl	a	accu contient longueur de chaîne
F5D2	4F	ld	c,a	
F5D3	06FF	ld	b,FF	moins longueur dans bc
F5D5	03	inc	bc	
F5D6	CDE6F5	call	F5E6	étendre zone de chaînes vers le bas
F5D9	D0	ret	nc	y a-t-il de la place ?
F5DA	CD3EFC	call	FC3E	non, déclencher Garbage Collection
F5DD	CDE6F5	call	F5E6	y a-t-il maintenant de la place ?
F5E0	D0	ret	nc	oui
F5E1	1E0E	ld	e,0E	'String space full'
F5E3	C394CA	jp	CA94	sortir message d'erreur

BASIC 1.0

***** y a-t-il de la place dans la zone des chaînes

F5E6	2A89AE	ld	hl,(AE89)	fin des tableaux
F5E9	EB	ex	de,hl	
F5EA	2A8DB0	ld	hl,(B08D)	début des chaînes
F5ED	09	add	hl,bc	moins longueur de la nouvelle chaîne
F5EE	CDB8FF	call	FFB8	comparer hl <> de
F5F1	D8	ret	c	
F5F2	228DB0	ld	(B08D),hl	début des chaînes
F5F5	23	inc	hl	
F5F6	EB	ex	de,hl	dans de
F5F7	C9	ret		

***** réserver place dans la zone des chaînes

F5F8	2A89AE	ld	hl,(AE89)	fin des tableaux
F5FB	C5	push	bc	contient nombre d'octets
F5FC	D5	push	de	
F5FD	D5	push	de	
F5FE	E5	push	hl	
F5FF	CD18F6	call	F618	y a-t-il de la place ?
F602	DA3EF7	jp	c,F73E	non, 'Memory full'
F605	E1	pop	hl	
F606	C1	pop	bc	
F607	D5	push	de	
F608	7D	ld	a,l	
F609	91	sub	c	
F60A	4F	ld	c,a	
F60B	7C	ld	a,h	
F60C	98	sbc	a,b	
F60D	47	ld	b,a	
F60E	2B	dec	hl	
F60F	1B	dec	de	
F610	B1	or	c	
F611	C4F5FF	call	nz,FFF5	laddr
F614	E1	pop	hl	
F615	D1	pop	de	
F616	C1	pop	bc	
F617	C9	ret		

***** Y a-t-il de la place dans la zone des chaînes

F618	09	add	hl,bc	fin tableaux plus nouvelle place
------	----	-----	-------	----------------------------------

BASIC 1.0

F619	D8	ret	c	dépassement ?
F61A	EB	ex	de,h1	
F61B	CD22F6	call	F622	comparer nouvelle fin des variables avec début des chaînes
F61E	D0	ret	nc	y a-t-il de la place ?
F61F	CD3EFC	call	FC3E	non, déclencher Garbage Collection
F622	2A8DB0	ld	h1,(B08D)	début des chaînes
F625	C3B8FF	jp	FFB8	comparer h1 <> de
*****				calculer place mémoire libre
F628	2A89AE	ld	h1,(AE89)	fin des variables
F62B	EB	ex	de,h1	
F62C	2A8DB0	ld	h1,(B08D)	début des chaînes
F62F	C3CFFF	jp	FFFC	h1 := h1 - de
*****				mettre en place le buffer d'entrée
F632	110100	ld	de,0001	
F635	1803	jr	F63A	
*****				mettre en place buffer de sortie
F637	110208	ld	de,0802	
F63A	C5	push	bc	
F63B	E5	push	h1	
F63C	2191B0	ld	h1,B091	
F63F	7E	ld	a,(h1)	
F640	B7	or	a	
F641	201D	jr	nz,F660	
F643	D5	push	de	
F644	E5	push	h1	
F645	210010	ld	h1,1000	
F648	010000	ld	bc,0000	
F64B	CD43F7	call	F743	
F64E	2292B0	ld	(B092),h1	nouvelle fin de la Ram libre
F651	EB	ex	de,h1	
F652	2A7DAE	ld	h1,(AE7D)	fin de la Ram libre
F655	2294B0	ld	(B094),h1	mémoire pour Ram libre
F658	EB	ex	de,h1	
F659	227DAE	ld	(AE7D),h1	fin de la Ram libre
F65C	E1	pop	h1	
F65D	D1	pop	de	

BASIC 1.0

```

F65E 3E04    ld    a,04
F660 B3      or    e
F661 77      ld    (hl),a
F662 2A92B0  ld    hl,(B092)  nouvelle fin de la Ram libre
F665 23      inc   hl
F666 1E00    ld    e,00
F668 19      add   hl,de
F669 EB      ex    de,hl
F66A E1      pop   hl
F66B C1      pop   bc
F66C C9      ret

```

***** fermer buffer d'entrée

```

F66D 3EFE    ld    a,FE
F66F 1806    jr    F677

```

***** fermer buffer de sortie

```

F671 3EFD    ld    a,FD
F673 1802    jr    F677

```

```

F675 3EFF    ld    a,FF
F677 C5      push  bc
F678 D5      push  de
F679 E5      push  hl
F67A 2191B0  ld    hl,B091
F67D A6      and   (hl)
F67E 77      ld    (hl),a
F67F FE04    cp    04
F681 2016    jr    nz,F699
F683 2A92B0  ld    hl,(B092)  nouvelle fin de la Ram libre
F686 EB      ex    de,hl
F687 210010  ld    hl,1000
F68A CD2EF7  call  F72E
F68D 200A    jr    nz,F699
F68F AF      xor   a
F690 3291B0  ld    (B091),a
F693 2A94B0  ld    hl,(B094)  mémoire pour Ram libre
F696 227DAE  ld    (AE7D),hl  fin de la Ram libre
F699 E1      pop   hl

```

BASIC 1.0

F69A D1 pop de
 F69B C1 pop bc
 F69C C9 ret

instruction Basic SYMBOL
 'AFTER'

F69D FE80 cp 80
 F69F 282C Jr z,F6CD
 F6A1 CD67CE call CE67 aller chercher valeurs 8 bits
 F6A4 4F ld c,a
 F6A5 CD37DD call DD37 tester si encore un caractère
 F6A8 2C db 2C ', '
 F6A9 0608 ld b,08 8 valeurs
 F6AB CD67CE call CE67 aller chercher valeurs 8 bits
 F6AE F5 push af sur pile
 F6AF 05 dec b
 F6B0 2808 Jr z,F6BA déjà 8 valeurs ?
 F6B2 CD55DD call DD55 virgule suit ?
 F6B5 38F4 Jr c,F6AB oui, aller chercher valeur suivante
 F6B7 AF xor a
 F6B8 18F4 Jr F6AE

F6BA EB ex de,h1 sauver h1
 F6BB 79 ld a,c caractère dans a
 F6BC CDA5BB call BBA5 TXT GET MATRIX
 F6BF 3068 Jr nc,F729 matrice pas dans Ram, 'Improper
 Argument'

F6C1 010800 ld bc,0008 8
 F6C4 09 add hl,bc plus adresse de matrice
 F6C5 F1 pop af aller chercher octet sur pile
 F6C6 2B dec hl
 F6C7 77 ld (hl),a écrire dans table de matrice
 F6C8 0D dec c octet suivant
 F6C9 20FA Jr nz,F6C5
 F6CB EB ex de,h1 ramener hl
 F6CC C9 ret

SYMBOL AFTER

F6CD CD3FDD call DD3F ignorer espaces
 F6D0 CD86CE call CE86 aller chercher valeur entière avec
 signe

BASIC 1.0

```

F6D3 E5      push  hl
F6D4 210001  ld     hl,0100      256
F6D7 CDB8FF  call  FFB8          comparer hl <> de
F6DA 384D      jr     c,F729        supérieur égal 256, 'Improper
                                argument'

F6DC D5      push  de
F6DD CDAEBB  call  BBAE          TXT GET M TABLE
F6E0 EB      ex     de,hl        adresse de matrice dans de
F6E1 301D      jr     nc,F700      matrice pas encore définie ?
F6E3 2F      cpl   a
F6E4 6F      ld     l,a
F6E5 2600     ld     h,00
F6E7 23      inc   hl
F6E8 29      add   hl,hl
F6E9 29      add   hl,hl
F6EA 29      add   hl,hl
F6EB 1B      dec   de
F6EC CD2EF7  call  F72E
F6EF 2038     jr     nz,F729      'Improper argument'
F6F1 2A96B0  ld     hl,(B096)
F6F4 227DAE  ld     (AE7D),hl   fin de la Ram libre
F6F7 CD75F6  call  F675
F6FA 110001  ld     de,0100
F6FD CDABBB  call  BBAB          TXT SET M TABLE
F700 D1      pop   de
F701 CD06F7  call  F706
F704 E1      pop   hl

```

BASIC 1.0

F705	C9	ret		
F706	AF	xor	a	
F707	93	sub	e	
F708	6F	ld	l,a	
F709	3E01	ld	a,01	
F70B	9A	sbc	a,d	
F70C	67	ld	h,a	
F70D	B5	or	l	
F70E	C8	ret	z	
F70F	D5	push	de	ranger premier caractère
F710	29	add	hl,hl	
F711	29	add	hl,hl	
F712	29	add	hl,hl	
F713	010040	ld	bc,4000	
F716	CD43F7	call	F743	
F719	EB	ex	de,hl	
F71A	2A7DAE	ld	hl,(AE7D)	fin de Ram libre
F71D	2296B0	ld	(B096),hl	
F720	EB	ex	de,hl	
F721	227DAE	ld	(AE7D),hl	fin de Ram libre
F724	D1	pop	de	premier caractère
F725	23	inc	hl	adresse de début de la table
F726	C3ABBB	jp	BBAB	TXT SET M TABLE
F729	1E05	ld	e,05	'Improper argument'
F72B	C394CA	jp	CA94	sortir message d'erreur
F72E	E5	push	hl	
F72F	2A7BAE	ld	hl,(AE7B)	HIMEM
F732	CDB8FF	call	FFB8	comparer hl <> de
F735	E1	pop	hl	
F736	C0	ret	nz	
F737	19	add	hl,de	
F738	227DAE	ld	(AE7D),hl	fin de Ram libre
F73B	EB	ex	de,hl	
F73C	1812	jr	F750	
F73E	1E07	ld	e,07	'Memory full'
F740	C394CA	jp	CA94	sortir message d'erreur

BASIC 1.0

```

F743 EB      ex      de,h1
F744 2A7BAE ld      hl,(AE7B) HIMEM
F747 CDCFFF call   FFCF      hl := hl - de
F74A CDBEFF call   FFBE      comparer hl <> bc
F74D 38EF    Jr      c,F73E 'Memory full'
F74F EB      ex      de,h1
F750 CD3EFC call   FC3E      Garbage Collection
F753 D5      push   de
F754 2A7DAE ld      hl,(AE7D) fin de Ram libre
F757 CDB8FF call   FFB8      comparer hl <> de
F75A 38E2    Jr      c,F73E 'Memory full'
F75C CD1DF5 call   F51D      calculer longueur zone des chaines
F75F 2A89AE ld      hl,(AE89) fin des tableaux
F762 09      add     hl,bc      plus longueur zone des chaines
F763 38D9    Jr      c,F73E 'Memory full'
F765 2B      dec     hl
F766 CDB8FF call   FFB8      comparer hl <> de
F769 30D3    Jr      nc,F73E 'Memory full'
F76B 2A7BAE ld      hl,(AE7B) HIMEM
F76E EB      ex      de,h1
F76F CDCFFF call   FFCF      hl := hl - de
F772 2298B0 ld      (B098),hl
F775 11BBF7 ld      de,F7BB
F778 CD74DA call   DA74
F77B ED4B98B0 ld     bc,(B098)
F77F 78      ld      a,b
F780 07      rlca
F781 3816    Jr      c,F799
F783 B1      or      c
F784 282F    Jr      z,F7B5
F786 2A8FB0 ld      hl,(B08F) fin des chaines
F789 54      ld      d,h
F78A 5D      ld      e,l
F78B 09      add     hl,bc
F78C E5      push   hl
F78D CD1DF5 call   F51D      calculer longueur zone des chaines
F790 EB      ex      de,h1
F791 78      ld      a,b
F792 B1      or      c
F793 C4F5FF call   nz,FFF5      lddr

```

BASIC 1.0

F796	E1	pop	h1
F797	1815	Jr	F7AE

BASIC 1.0

```

F799 2A8DB0 ld hl,(B08D) début des chaînes
F79C 54 ld d,h
F79D 5D ld e,l
F79E 09 add hl,bc
F79F E5 push hl
F7A0 CD1DF5 call F51D calculer longueur zone des chaînes
F7A3 EB ex de,hl
F7A4 23 inc hl
F7A5 13 inc de
F7A6 78 ld a,b
F7A7 B1 or c
F7A8 C4F2FF call nz,FFF2 ldir
F7AB EB ex de,hl
F7AC 2B dec hl
F7AD D1 pop de
F7AE 228FB0 ld (B08F),hl fin des chaînes
F7B1 EB ex de,hl
F7B2 228DB0 ld (B08D),hl début des chaînes
F7B5 E1 pop hl
F7B6 227BAE ld (AE7B),hl HIMEM
F7B9 AF xor a
F7BA C9 ret

F7BB 2A83AE ld hl,(AE83) fin du programme
F7BE CDBEFF call FFBE comparer hl <> bc
F7C1 D0 ret nc
F7C2 2A98B0 ld hl,(B098)
F7C5 09 add hl,bc
F7C6 EB ex de,hl
F7C7 72 ld (hl),d
F7C8 2B dec hl
F7C9 73 ld (hl),e
F7CA C9 ret

***** lire chaîne
F7CB 23 inc hl
F7CC CDF9F7 call F7F9
F7CF 7E ld a,(hl)
F7D0 FE22 cp 22 ''' , fin de chaîne ?
F7D2 CA3FDD jp z,DD3F oui, ignorer espaces suivants

```

BASIC 1.0

```
F7D5 B7      or      a
F7D6 2837   jr      z,F80F
F7D8 04     inc     b
F7D9 23     inc     hl
F7DA 18F3   jr      F7CF
```

```
F7DC CDF9F7  call   F7F9
F7DF 7E     ld      a,(hl)
F7E0 B7     or      a
F7E1 C8     ret     z
F7E2 23     inc     hl
F7E3 04     inc     b
F7E4 18F9   jr      F7DF
```

```
F7E6 CDF9F7  call   F7F9
F7E9 4F     ld      c,a
F7EA 7E     ld      a,(hl)
F7EB B7     or      a
F7EC 2821   jr      z,F80F
F7EE B9     cp      c
F7EF 281E   jr      z,F80F
F7F1 FE2C   cp      2C      ', '
F7F3 281A   jr      z,F80F
F7F5 23     inc     hl
F7F6 04     inc     b
F7F7 18F1   jr      F7EA
```

```
F7F9 D1      pop     de
F7FA E5      push    hl
F7FB 0600   ld      b,00
F7FD CDFBFF  call   FFFB      jp (de)
F800 D1      pop     de
F801 E5      push    hl
F802 21BAB0  ld      hl,BOBA   pointeur sur pile du descripteur
F805 70     ld      (hl),b    longueur
F806 23     inc     hl
F807 73     ld      (hl),e
F808 23     inc     hl         adresse
F809 72     ld      (hl),d
```

BASIC 1.0

```

F80A CDBAFB call FBBA
F80D E1 pop hl
F80E C9 ret

F80F E5 push hl
F810 04 inc b
F811 05 dec b
F812 2812 jr z,F826
F814 2B dec hl
F815 7E ld a,(hl)
F816 FE20 cp 20 '5'
F818 28F7 jr z,F811
F81A FE09 cp 09 TAB
F81C 28F3 jr z,F811
F81E FE0D cp 0D CR
F820 28EF jr z,F811
F822 FE0A cp 0A LF
F824 28EB jr z,F811
F826 E1 pop hl
F827 C9 ret

```

```

F828 CDDAFB call FBDA sortir chaîne
F82B C8 ret z aller chercher paramètres de chaîne
F82C 1A ld a,(de) chaîne vide ?
F82D 13 inc de aller chercher caractère
F82E CD6EC3 call C36E augmenter pointeur
F831 10F9 djnz F82C sortir caractère
F833 C9 ret caractère suivant

```

```

F834 0139F8 ld bc,F839 fonction Basic LOWER$
F837 180C jr F845 convertir majuscules en minuscules

```

```

F839 FE41 cp 41 conversion majuscules en minuscules
F83B D8 ret c 'A'
F83C FE5B cp 5B 'Z'+1
F83E D0 ret nc
F83F C620 add a,20 'a'-'A'

```

BASIC 1.0

F841 C9 ret

fonction Basic UPPER\$
convertir majusc. en minusc.

F842 018AFF ld bc,FF8A
F845 C5 push bc
F846 2AC2B0 ld hl,(BOC2)
F849 7E ld a,(hl)
F84A CD19FC call FC19

longueur de chaîne
réserver place, placer descripteur
de chaîne

F84D D5 push de
F84E CDDAFB call FBDA
F851 E1 pop hl
F852 C1 pop _bc
F853 3C inc a
F854 3D dec a
F855 CABAFB jp z,FBBA
F858 F5 push af
F859 1A ld a,(de)
F85A 13 inc de
F85B CDF9FF call FFF9
F85E 77 ld (hl),a
F85F 23 inc hl
F860 F1 pop af
F861 18F1 jr F854

aller chercher paramètres de chaîne

jp (bc), exécuter conversion

addition de chaîne

F863 E5 push hl
F864 7E ld a,(hl)
F865 2AC2B0 ld hl,(BOC2)
F868 86 add a,(hl)
F869 1E0F ld e,OF
F86B DA94CA jp c,CA94

longueur de chaîne
plus longueur de deuxième chaîne
'String too long'
sortir message d'erreur

F86E CD19FC call FC19

réserver place, placer descripteur
de chaîne

F871 E1 pop hl
F872 D5 push de
F873 E5 push hl
F874 CDDAFB call FBDA
F877 48 ld c,b

aller chercher paramètres de chaîne

BASIC 1.0

```

F878 EB      ex      de,hl
F879 E3      ex      (sp),hl
F87A CDE8FB  call    FBE8
F87D E1      pop     hl
F87E E3      ex      (sp),hl
F87F 78      ld      a,b
F880 CD8BF8  call    F88B
F883 D1      pop     de
F884 79      ld      a,c
F885 CD8BF8  call    F88B
F888 C3BAFB  jp      FBBA

F88B C5      push   bc
F88C EB      ex      de,hl
F88D 4F      ld      c,a
F88E 0600    ld      b,00
F890 B7      or      a
F891 C4F2FF  call    nz,FFF2      ldir
F894 EB      ex      de,hl
F895 C1      pop     bc
F896 C9      ret

```

comparaison de chaînes

```

F897 E5      push   hl
F898 CDDAFB  call    FBDA      aller chercher paramètres de chaîne
F89B 48      ld      c,b
F89C E1      pop     hl
F89D D5      push   de
F89E CDE8FB  call    FBE8
F8A1 E1      pop     hl
F8A2 78      ld      a,b
F8A3 B1      or      c
F8A4 C8      ret     z
F8A5 79      ld      a,c
F8A6 B7      or      a
F8A7 280C    jr     z,F8B5
F8A9 78      ld      a,b
F8AA B7      or      a
F8AB 2809    jr     z,F8B6
F8AD 05      dec    b

```

BASIC 1.0

F8AE	0D	dec	c	
F8AF	1A	ld	a,(de)	comparer caractère première chaîne
F8B0	13	inc	de	
F8B1	BE	cp	(hl)	avec seconde chaîne
F8B2	23	inc	hl	
F8B3	28ED	jr	z,F8A2	identique, alors continuer compar.
F8B5	3F	ccf		
F8B6	9F	sbc	a,a	fixer flags pour résultat
F8B7	C0	ret	nz	
F8B8	3C	inc	a	
F8B9	C9	ret		

***** fonction Basic BIN\$

F8BA	CDCEF8	call	F8CE	aller chercher arguments
F8BD	D5	push	de	
F8BE	CD14F1	call	F114	convertir en chaîne binaire
F8C1	EB	ex	de,hl	
F8C2	185E	jr	F922	accepter chaîne

***** fonction Basic HEX\$

F8C4	CDCEF8	call	F8CE	aller chercher arguments
F8C7	D5	push	de	
F8C8	CD19F1	call	F119	convertir en chaîne hexa
F8CB	EB	ex	de,hl	
F8CC	1854	jr	F92	accepter chaîne

***** aller chercher argument pour BIN\$ et HEX\$

F8CE	CDFBCE	call	CEFB	aller chercher expression
F8D1	CD53FF	call	FF53	et placer sur pile Basic
F8D4	CD55DD	call	DD55	virgule suit ?
F8D7	9F	sbc	a,a	0 comme défaut
F8D8	DC67CE	call	c.CE67	oui, aller chercher valeur 8 bits
F8DB	FE11	cp	11	supérieur égal 17 ?
F8DD	D29CFA	jp	nc,FA9C	'Improper argument'
F8E0	47	ld	b,a	
F8E1	CD37DD	call	DD37	tester si encore un caractère
F8E4	29	db	29	')'
F8E5	EB	ex	hl,de	
F8E6	79	ld	a,c	
F8E7	C3A0F5	jp	F5A0	libérer place dans pile Basic

BASIC 1.0

```

*****
F8EA CD37DD call DD37 fonction Basic DEC$
F8ED 28 db 28 tester si encore un caractère
F8EE CDFBCE call CEFB '(, déjà produit avec appel fonction
F8F1 CD37DD call DD37 aller chercher expression
F8F4 2C db 2C tester si encore un caractère
F8F5 CD53FF call FF53 ',
F8F8 CD9FCE call CE9F et placer sur pile Basic
                                aller chercher expression et
                                paramètres chaîne
F8FB CD37DD call DD37 tester si encore un caractère
F8FE 29 db 29 ')
F8FF E5 push hl
F900 79 ld a,c longueur
F901 CDA0F5 call F5A0 libérer place dans pile Basic
F904 D5 push de
F905 79 ld a,c longueur
F906 CD4BFF call FF4B accepter variable
F909 D1 pop de
F90A 78 ld a,b
F90B B7 or a
F90C C4BAF3 call nz,F3BA tester si caractère de formatage
F90F 300A jr nc,F91B 'Improper argument'
F911 78 ld a,b
F912 B7 or a
F913 2006 jr nz,F91B 'Improper argument'
F915 79 ld a,c
F916 CD9FEE call EE9F formater nombre
F919 1807 jr F922 accepter chaîne

F91B C39CFA jp FA9C 'Improper argument'

*****
F91E E5 push hl fonction Basic STR$
F91F CD9DEE call EE9D convertir nombre en chaîne
F922 E5 push hl
F923 01FFFF ld bc,FFFF compteur pour longueur de chaîne sur
                                -1
F926 03 inc bc augmenter compteur
F927 7E ld a,(hl) aller chercher caractère
F928 23 inc hl

```

BASIC 1.0

F929	B7	or	a	octet nul ?
F92A	20FA	Jr	nz,F926	non, prochain caractère
F92C	79	ld	a,c	longueur de chaîne dans a
F92D	CD19FC	call	FC19	réserver place, placer descripteur de chaîne
F930	E1	pop	hl	
F931	B7	or	a	
F932	D5	push	de	
F933	C4F2FF	call	nz,FFF2	ldir
F936	D1	pop	de	
F937	CDBAFB	call	FBBA	
F93A	E1	pop	hl	
F93B	C9	ret		
*****				fonction Basic LEFT\$
F93C	CDE9F9	call	F9E9	amener chaîne et nombre 8 bits
F93F	0E00	ld	c,00	à partir de position 0
F941	182A	Jr	F96D	
*****				fonction Basic RIGHT\$
F943	CDE9F9	call	F9E9	amener chaîne et nombre 8 bits
F946	1A	ld	a,(de)	longueur de chaîne
F947	90	sub	b	moins paramètre
F948	4F	ld	c,a	donne position de départ
F949	1822	Jr	F96D	
*****				fonction Basic MID\$
F94B	CD37DD	call	DD37	tester si encore un caractère
F94E	28	db	28	'('
F94F	CDE9F9	call	F9E9	amener chaîne et nombre 8 bits
F952	78	ld	a,b	
F953	B7	or	a	
F954	CA9CFA	Jp	z,FA9C	'Improper argument'
F957	05	dec	b	
F958	48	ld	c,b	
F959	D5	push	de	
F95A	C5	push	bc	
F95B	CDFBF9	call	F9FB	aller chercher 3ème argument (défaut = 255)
F95E	C1	pop	bc	

BASIC 1.0

```

F95F E3      ex      (sp),hl
F960 7E      ld      a,(hl)
F961 91      sub     c
F962 0600    ld      b,00
F964 3805    jr      c,F96B
F966 BB      cp      e
F967 47      ld      b,a
F968 3801    jr      c,F96B
F96A 43      ld      b,e
F96B EB      ex      de,hl
F96C E1      pop     hl
F96D CD37DD  call   DD37      tester si encore un caractère
F970 29      db      29      ')'
F971 E5      push    hl
F972 EB      ex      de,hl
F973 7E      ld      a,(hl)
F974 B8      cp      b
F975 78      ld      a,b
F976 3003    jr      nc,F97B
F978 7E      ld      a,(hl)
F979 0E00    ld      c,00
F97B F5      push    af
F97C CD19FC  call   FC19      réserver place, placer descripteur
                    de chaîne

F97F D5      push    de
F980 CDE8FB  call   FBE8
F983 EB      ex      de,hl
F984 D1      pop     de
F985 0600    ld      b,00
F987 09      add     hl,bc
F988 F1      pop     af
F989 4F      ld      c,a
F98A B7      or      a
F98B C4F2FF  call   nz,FFF2    ldir
F98E CDBAFB  call   FBBA
F991 E1      pop     hl
F992 C9      ret

```

```

***** instruction Basic MID$
F993 CD37DD  call   DD37      tester si encore un caractère

```

BASIC 1.0

F996	28	db	28	'('
F998	CD86D6	call	D686	aller chercher variable
F99A	CD3CFF	call	FF3C	type 'chaîne', sinon 'Type mismatch'
F99D	E5	push	hl	
F99E	EB	ex	de,hl	
F99F	CD21FB	call	FB21	
F9A2	E3	ex	(sp),hl	
F9A3	CD37DD	call	DD37	tester si encore un caractère
F9A6	2C	db	2C	','
F9A7	CD6DCE	call	CE6D	aller chercher valeur 8 bits non nulle
F9AA	47	ld	b,a	
F9AB	CDFBF9	call	F9FB	aller chercher 3ème argument (défaut = 255)
F9AE	4B	ld	c,e	
F9AF	CD37DD	call	DD37	tester si encore un caractère
F9B2	29	db	29	')
F9B3	CD37DD	call	DD37	tester si encore un caractère
F9B6	EF	db	EF	'='
F9B7	C5	push	bc	
F9B8	CD9FCE	call	CE9F	aller chercher expression et paramètres de chaîne
F9BB	78	ld	a,b	
F9BC	C1	pop	bc	
F9BD	E3	ex	(sp),hl	
F9BE	0C	inc	c	
F9BF	0D	dec	c	
F9C0	2825	Jr	z,F9E7	
F9C2	F5	push	af	
F9C3	7E	ld	a,(hl)	
F9C4	90	sub	b	
F9C5	DA9CFA	jp	c,FA9C	'Improper argument'
F9C8	3C	inc	a	
F9C9	B9	cp	c	
F9CA	3801	Jr	c,F9CD	
F9CC	79	ld	a,c	
F9CD	4F	ld	c,a	
F9CE	78	ld	a,b	
F9CF	3D	dec	a	
F9D0	23	inc	hl	

BASIC 1.0

```

F9D1 86      add    a,(h1)
F9D2 23      inc    hl
F9D3 66      ld     h,(h1)
F9D4 6F      ld     l,a
F9D5 8C      adc    a,h
F9D6 95      sub    l
F9D7 67      ld     h,a
F9D8 F1      pop    af
F9D9 47      ld     b,a
F9DA EB      ex     de,hl
F9DB 79      ld     a,c
F9DC B8      cp     b
F9DD 3801    jr     c,F9E0
F9DF 78      ld     a,b
F9E0 4F      ld     c,a
F9E1 0600    ld     b,00
F9E3 B7      or     a
F9E4 C4F2FF  call   nz,FFF2    ldir
F9E7 E1      pop    hl
F9E8 C9      ret

```

```

*****          amener chaîne et valeur 8 bits
F9E9 CDA5CE  call   CEA5      amener expression chaîne
F9EC CD37DD  call   DD37      tester si encore un caractère
F9EF 2C      db     2C      ','
F9F0 E5      push   hl
F9F1 2AC2B0  ld     hl,(BOC2)
F9F4 E3      ex     (sp),hl
F9F5 CD67CE  call   CE67      amener valeur 8 bits
F9F8 47      ld     b,a
F9F9 D1      pop    de
F9FA C9      ret

```

```

*****          amener 3ème argument pour MID$
F9FB 1EFF    ld     e,FF      défaut 255
F9FD 7E      ld     a,(hl)
F9FE FE29    cp     29      ')'
FA00 C8      ret    z
FA01 CD37DD  call   DD37      tester si encore un caractère
FA04 2C      db     2C      ','

```

BASIC 1.0

FA05	CD67CE	call	CE67	amener valeur 8 bits
FA08	5F	ld	e,a	
FA09	C9	ret		
*****				fonction Basic LEN
FA0A	CDDAFB	call	FBDA	amener paramètres de chaîne,
				longueur dans a
FA0D	C30AFF	jp	FF0A	accepter contenu accu comme nombre
				entier
*****				fonction Basic ASC
FA10	CD70FA	call	FA70	code ASCII du premier caractère
FA13	C30AFF	jp	- FF0A	accepter contenu accu comme nombre
				entier
*****				fonction Basic CHR\$
FA16	CD92FA	call	FA92	CINT, < 256
FA19	F5	push	af	
FA1A	3E01	ld	a,01	longueur 1
FA1C	CD19FC	call	FC19	réserver place, placer descripteur
FA1F	F1	pop	af	
FA20	12	ld	(de),a	placer code ASCII comme chaîne
FA21	C3BAFB	jp	FBBA	
*****				INKEY\$
FA24	E5	push	hl	
FA25	CD2AFA	call	FA2A	
FA28	E1	pop	hl	
FA29	C9	ret		
FA2A	CD39C4	call	C439	KM READ CHAR
FA2D	38EA	Jr	c,FA19	touche enfoncée ?
FA2F	AF	xor	a	non
FA30	32BAB0	ld	(BOBA),a	descripteur de chaîne, longueur
FA33	C3BAFB	jp	FBBA	
*****				STRING\$
FA36	CD67CE	call	CE67	amener valeur 8 bits, longueur
FA39	4F	ld	c,a	
FA3A	CD37DD	call	DD37	tester si encore un caractère

BASIC 1.0

FA3D	2C	db	2C	' , '
FA3E	CDFBCE	call	CEFB	aller chercher expression
FA41	CD37DD	call	DD37	tester si encore un caractère
FA44	29	db	29	')'
FA45	E5	push	hl	
FA46	CD45FF	call	FF45	tester si chaîne
FA49	2805	jr	z,FA50	oui
FA4B	CD92FA	call	FA92	CINT, < 256
FA4E	1803	jr	FA53	
FA50	CD70FA	call	FA70	amener code ASCII du premier caractère
FA53	41	ld	b,c	
FA54	4F	ld	c,a	
FA55	1807	jr	FA5E	
*****				fonction Basic SPACE\$
FA57	CD92FA	call	FA92	CINT, < 256
FA5A	47	ld	b,a	
FA5B	0E20	ld	c,20	'5'
FA5D	E5	push	hl	
FA5E	78	ld	a,b	
FA5F	CD19FC	call	FC19	réserver place, placer descripteur
FA62	04	inc	b	
FA63	05	dec	b	
FA64	2805	jr	z,FA6B	
FA66	79	ld	a,c	
FA67	12	ld	(de),a	
FA68	13	inc	de	
FA69	18F8	jr	FA63	
FA6B	CDBAFB	call	FBBA	
FA6E	E1	pop	hl	
FA6F	C9	ret		
*****				amener code ASCII
FA70	CDDAFB	call	FBDA	amener paramètres de chaîne
FA73	2827	jr	z,FA9C	chaîne vide, 'Improper argument'
FA75	1A	ld	a,(de)	code du premier caractère
FA76	C9	ret		

BASIC 1.0

```

*****
FA77 CDDAFB call FBDA          fonction Basic VAL
FA7A CA0AFF jp z,FFOA         amener parametère de chaîne
FA7D EB ex de,h1             chaîne vide, alors zéro
FA7E E5 push hl
FA7F 5F ld e,a
FA80 1600 ld d,00
FA82 19 add hl,de
FA83 5E ld e,(hl)
FA84 72 ld (hl),d
FA85 E3 ex (sp),hl
FA86 D5 push de
FA87 CDA3EC call ECA3
FA8A D1 pop de
FA8B E1 pop hl
FA8C 73 ld (hl),e
FA8D D8 ret c
FA8E 1E0D ld e,0D           'Type mismatch'
FA90 180C jr FA9E           sortir message d'erreur

*****
FA92 E5 push hl
FA93 CD8DFE call FE8D        CINT
FA96 EB ex de,h1
FA97 E1 pop hl
FA98 7A ld a,d             Hi-Byte
FA99 B7 or a               zéro ?
FA9A 7B ld a,e             charger Lo-Byte
FA9B C8 ret z
FA9C 1E05 ld e,05         'Improper Argument'
FA9E C394CA jp CA94       sortir message d'erreur

*****
FAA1 CDFBCE call CEFB        fonction Basic INSTR
FAA4 CD45FF call FF45        aller chercher expression
FAA7 0E01 ld c,01         tester si chaîne
FAA9 280F jr z,FABA       position de départ défaut 1
FAAB CD92FA call FA92        CINT, < 256
FAAE B7 or a
FAAF CA9CFA jp z,FA9C     'Improper argument'

```

BASIC 1.0

FAB2	4F	ld	c,a	
FAB3	CD37DD	call	DD37	tester si encore un caractère
FAB6	2C	db	2C	','
FAB7	CDA5CE	call	CEA5	amener expression chaîne
FABA	CD37DD	call	DD37	tester si encore un caractère
FABD	2C	db	2C	','
FABE	E5	push	h1	
FABF	2AC2B0	ld	h1,(BOC2)	
FAC2	E3	ex	(sp),h1	
FAC3	CD9FCE	call	CE9F	amener expression et paramètres chaîne
FAC6	CD37DD	call	DD37	tester si encore un caractère
FAC9	29	db	29	')
FACA	E3	ex	(sp),h1	
FACB	79	ld	a,c	
FACC	CDD4FA	call	FAD4	
FACF	CDOAFF	call	FFOA	accepter contenu accu comme nombre entier
FAD2	E1	pop	h1	
FAD3	C9	ret		
FAD4	F5	push	af	
FAD5	48	ld	c,b	
FAD6	D5	push	de	
FAD7	CDE8FB	call	FBE8	
FADA	E1	pop	h1	
FADB	F1	pop	af	
FADC	E5	push	h1	
FADD	6F	ld	l,a	
FADE	60	ld	h,b	
FADF	78	ld	a,b	
FAE0	BD	cp	l	
FAE1	382D	jr	c,FB10	
FAE3	2D	dec	l	
FAE4	7D	ld	a,l	
FAE5	83	add	a,e	
FAE6	5F	ld	e,a	
FAE7	8A	adc	a,d	
FAE8	93	sub	e	
FAE9	57	ld	d,a	

BASIC 1.0

FAEA	78	ld	a,b
FAEB	95	sub	l
FAEC	47	ld	b,a
FAED	79	ld	a,c
FAEE	D601	sub	01
FAFO	7D	ld	a,l
FAF1	3C	inc	a
FAF2	381D	jr	c,FB11
FAF4	E3	ex	(sp),hl
FAF5	C5	push	bc
FAF6	D5	push	de
FAF7	E5	push	hl
FAF8	1A	ld	a,(de)
FAF9	BE	cp	(hl)
FAFA	200D	jr	nz,FB09
FAFC	23	inc	hl
FAFD	0D	dec	c
FAFE	2813	jr	z,FB13
FB00	13	inc	de
FB01	05	dec	b
FB02	20F4	jr	nz,FAF8
FB04	E1	pop	hl
FB05	D1	pop	de
FB06	C1	pop	bc
FB07	1807	Jr	FB10
FB09	E1	pop	hl
FB0A	D1	pop	de
FB0B	C1	pop	bc
FB0C	13	inc	de
FB0D	05	dec	b
FB0E	20E5	jr	nz,FAF5
FB10	AF	xor	a
FB11	D1	pop	de
FB12	C9	ret	
FB13	E1	pop	hl
FB14	D1	pop	de
FB15	C1	pop	bc
FB16	E1	pop	hl

BASIC 1.0

FB17	7C	ld	a,h	
FB18	90	sub	b	
FB19	3C	inc	a	
FB1A	C9	ret		
FB1B	112EFB	ld	de,FB2E	
FB1E	C374DA	jp	DA74	
FB21	E5	push	hl	
FB22	7E	ld	a,(hl)	
FB23	23	inc	hl	
FB24	4E	ld	c,(hl)	
FB25	23	inc	hl	
FB26	46	ld	b,(hl)	
FB27	EB	ex	de,hl	
FB28	B7	or	a	
FB29	C42EFB	call	nz,FB2E	
FB2C	E1	pop	hl	
FB2D	C9	ret		
FB2E	2A8DB0	ld	hl,(B08D)	début des chaînes
FB31	CDBEFF	call	FFBE	comparer hl <> bc
FB34	3007	jr	nc,FB3D	
FB36	2A8FB0	ld	hl,(B08F)	fin des chaînes
FB39	CDBEFF	call	FFBE	comparer hl <> bc
FB3C	D0	ret	nc	
FB3D	EB	ex	de,hl	
FB3E	2B	dec	hl	
FB3F	2B	dec	hl	
FB40	E5	push	hl	
FB41	CD8FFB	call	FB8F	
FB44	EB	ex	de,hl	
FB45	E1	pop	hl	
FB46	C3A6FB	jp	FBA6	descripteur de chaîne de (de) dans (hl)
FB49	2AC2B0	ld	hl,(B0C2)	
FB4C	11BAB0	ld	de,BOBA	descripteur de chaîne
FB4F	CDB8FF	call	FFB8	comparer hl <> de
FB52	D8	ret	c	

BASIC 1.0

FB53 CD8FFB call FB8F
 FB56 C3BAFB jp FBBA

FB59	2AC2B0	ld	hl,(B0C2)	pointeur sur descripteur de chaîne
FB5C	E5	push	hl	
FB5D	7E	ld	a,(hl)	longueur de chaîne
FB5E	B7	or	a	
FB5F	2826	jr	z,FB87	chaîne vide ?
FB61	23	inc	hl	
FB62	5E	ld	e,(hl)	
FB63	23	inc	hl	longueur de chaîne dans de
FB64	56	ld	d,(hl)	
FB65	2A81AE	ld	hl,(AE81)	début de programme
FB68	CDB8FF	call	FFB8	comparer hl <> de
FB6B	301E	jr	nc,FB8B	chaîne avant le programme
FB6D	2A8FB0	ld	hl,(B08F)	fin des chaînes
FB70	CDB8FF	call	FFB8	comparer hl <> de
FB73	3816	jr	c,FB8B	chaîne en dehors de zone des chaînes
FB75	2A83AE	ld	hl,(AE83)	fin du programme
FB78	CDB8FF	call	FFB8	comparer hl <> de
FB7B	300A	jr	nc,FB87	chaîne dans programme
FB7D	E1	pop	hl	
FB7E	E5	push	hl	
FB7F	119CB0	ld	de,B09C	
FB82	CDB8FF	call	FFB8	comparer hl <> de
FB85	2004	jr	nz,FB8B	
FB87	E1	pop	hl	
FB88	C3FFFB	jp	FBFF	
FB8B	E1	pop	hl	
FB8C	CDFFFB	call	FBFF	
FB8F	7E	ld	a,(hl)	
FB90	CD19FC	call	FC19	réserver place, placer descripteur
FB93	D5	push	de	
FB94	4E	ld	c,(hl)	longueur de chaîne dans c
FB95	0600	ld	b,00	Hi-Byte longueur zéro
FB97	23	inc	hl	
FB98	7E	ld	a,(hl)	
FB99	23	inc	hl	adresse de chaîne dans hl

BASIC 1.0

```

FB9A 66      ld      h,(h1)
FB9B 6F      ld      l,a
FB9C 78      ld      a,b
FB9D B1      or       c
FB9E C4F2FF  call    nz,FFF2      ldir, transférer chaîne
FBA1 D1      pop     de
FBA2 21BAB0  ld      hl,BOBA     descripteur de chaîne
FBA5 C9      ret

```

***** descripteur de chaîne de (de) dans (hl)

```

FBA6 1A      ld      a,(de)
FBA7 13      inc     de
FBA8 77      ld      (hl),a
FBA9 23      inc     hl
FBAA 1A      ld      a,(de)
FBAB 13      inc     de
FBAC 77      ld      (hl),a
FBAD 23      inc     hl
FBAE 1A      ld      a,(de)
FBAF 13      inc     de
FBB0 77      ld      (hl),a
FBB1 23      inc     hl
FBB2 C9      ret

```

***** initialiser pile du descripteur

```

FBB3 219CB0  ld      hl,B09C
FBB6 229AB0  ld      (B09A),hl   pointeur sur pile descripteur pour
                        chaînes
FBB9 C9      ret

```

```

FBBA 3E03     ld      a,03         'chaîne'
FBBC 32C1B0  ld      (BOC1),a    comme type de variable
FBBF 2A9AB0  ld      hl,(B09A)   pointeur dans pile descripteur
FBC2 22C2B0  ld      (BOC2),hl
FBC5 11BAB0  ld      de,BOBA     descripteur de chaîne
FBC8 CDB8FF  call    FFB8        comparer hl <> de
FBCB 1E10     ld      e,10        'String expression too complex'
FBCD CA94CA  jp      z,CA94      sortir message d'erreur

```

BASIC 1.0

FBDO	11BAB0	ld	de,BOBA	descripteur de chaîne
FBD3	CDAGFB	call	FBA6	descripteur de chaîne de (de) dans (hl)
FBD6	229AB0	ld	(B09A),hl	pointeur dans pile descripteur
FBD9	C9	ret		
*****				amener paramètres chaîne
FBDA	E5	push	hl	
FBDB	CD3CFF	call	FF3C	type 'chaîne', sinon 'Type mismatch'
FBDE	2AC2B0	ld	hl,(BOC2)	adresse du descripteur de chaînes
FBE1	CDE8FB	call	FBE8	
FBE4	E1	pop	hl	
FBE5	78	ld	a,b	longueur dans a et b, adresse dans de
FBE6	B7	or	a	
FBE7	C9	ret		
FBE8	CDFFFB	call	FBFF	
FBEB	C0	ret	nz	
FBEC	D5	push	de	
FBED	1B	dec	de	
FBEE	2A8DB0	ld	hl,(B08D)	début des chaînes
FBF1	CDB8FF	call	FFB8	comparer hl <> de
FBF4	2007	Jr	nz,FBFD	
FBF6	58	ld	e,b	
FBF7	1600	ld	d,00	
FBF9	19	add	hl,de	
FBFA	228DB0	ld	(B08D),hl	début des chaînes
FBFD	D1	pop	de	
FBFE	C9	ret		
FBFF	E5	push	hl	
FC00	46	ld	b,(hl)	
FC01	23	inc	hl	
FC02	7E	ld	a,(hl)	
FC03	23	inc	hl	
FC04	66	ld	h,(hl)	
FC05	6F	ld	l,a	
FC06	E3	ex	(sp),hl	
FC07	EB	ex	de,hl	

BASIC 1.0

FC08	2A9AB0	ld	hl,(B09A)	pointeur sur pile descripteur
FC0B	2B	dec	hl	
FC0C	2B	dec	hl	
FC0D	2B	dec	hl	
FC0E	CDB8FF	call	FFB8	comparer hl <> de
FC11	2003	jr	nz,FC16	
FC13	229AB0	ld	(B09A),hl	pointeur sur pile descripteur
FC16	EB	ex	de,hl	
FC17	D1	pop	de	
FC18	C9	ret		
*****				réserver place, placer descripteur
FC19	F5	push	af	
FC1A	C5	push	bc	
FC1B	E5	push	hl	
FC1C	F5	push	af	longueur de chaîne
FC1D	CDD1F5	call	F5D1	réserver place dans zone chaînes
FC20	F1	pop	af	
FC21	21BAB0	ld	hl,B0BA	descripteur de chaîne
FC24	77	ld	(hl),a	longueur de chaîne
FC25	23	inc	hl	
FC26	73	ld	(hl),e	
FC27	23	inc	hl	adresse de chaîne
FC28	72	ld	(hl),d	
FC29	E1	pop	hl	
FC2A	C1	pop	bc	
FC2B	F1	pop	af	
FC2C	C9	ret		
*****				fonction Basic FRE
FC2D	CD45FF	call	FF45	tester si chaîne
FC30	2006	jr	nz,FC38	non
FC32	CDDAFB	call	FBDA	
FC35	CD3EFC	call	FC3E	Garbage Collection
FC38	CD28F6	call	F628	calculer place libre en mémoire
FC3B	C360FE	jp	FE60	
*****				Garbage Collection
FC3E	C5	push	bc	
FC3F	D5	push	de	

BASIC 1.0

FC40	E5	push	hl	
FC41	2A8FB0	ld	hl,(B08F)	fin des chaînes
FC44	228DB0	ld	(B08D),hl	début des chaînes
FC47	210000	ld	hl,0000	
FC4A	22BDB0	ld	(B0BD),hl	
FC4D	2A89AE	ld	hl,(AE89)	fin des tableaux
FC50	22BF80	ld	(B0BF),hl	
FC53	CD7BFC	call	FC7B	
FC56	2ABDB0	ld	hl,(B0BD)	
FC59	7C	ld	a,h	
FC5A	B5	or	l	
FC5B	281A	jr	z,FC77	
FC5D	56	ld	d,(hl)	
FC5E	2B	dec	hl	
FC5F	5E	ld	e,(hl)	
FC60	E5	push	hl	
FC61	2B	dec	hl	
FC62	4E	ld	c,(hl)	
FC63	0600	ld	b,00	
FC65	2A8DB0	ld	hl,(B08D)	début des chaînes
FC68	EB	ex	de,hl	
FC69	09	add	hl,bc	
FC6A	2B	dec	hl	
FC6B	CDF5FF	call	FFF5	lddr
FC6E	13	inc	de	
FC6F	E1	pop	hl	
FC70	73	ld	(hl),e	
FC71	23	inc	hl	
FC72	72	ld	(hl),d	
FC73	1B	dec	de	
FC74	EB	ex	de,hl	
FC75	18CD	jr	FC44	
FC77	E1	pop	hl	
FC78	D1	pop	de	
FC79	C1	pop	bc	
FC7A	C9	ret		
FC7B	219CB0	ld	hl,B09C	
FC7E	ED5B9AB0	ld	de,(B09A)	pointeur sur pile descripteur

BASIC 1.0

```

FC82 CDB8FF call FFB8      comparer h1 <> de
FC85 280F   Jr    z,FC96
FC87 7E     ld    a,(h1)
FC88 23     inc   hl
FC89 4E     ld    c,(h1)
FC8A 23     inc   hl
FC8B 46     ld    b,(h1)
FC8C E5     push  hl
FC8D EB     ex    de,hl
FC8E B7     or    a
FC8F C49CFC call  nz,FC9C
FC92 E1     pop   hl
FC93 23     inc   hl
FC94 18E8   Jr    FC7E

FC96 119CFC ld    de,FC9C
FC99 C374DA  Jp    DA74

FC9C 2A8DB0 ld    hl,(B08D)  début des chaînes
FC9F CDBEFF call  FFBE      comparer h1 <> bc
FCA2 D8     ret    c
FCA3 2ABFBO ld    hl,(BOBF)
FCA6 CDBEFF call  FFBE      comparer h1 <> bc
FCA9 D0     ret    nc
FCAA EB     ex    de,hl
FCAB 22BDB0 ld    (BOBD),hl
FCAE ED43BFBO ld    (BOBF),bc
FCB2 C9     ret

*****
FCB3 CD2DFF call  FF2D      amener résultat numérique
FCB6 D252BD  Jp    nc,BD52   virgule flottante
FCB9 CDA3BD  call  BDA3      entier
FCBC 22C2B0 ld    (BOC2),hl
FCBF 21C3B0 ld    hl,BOC3
FCC2 C9     ret

FCC3 CDC2FE call  FEC2      UNT
FCC6 21C3B0 ld    hl,BOC3
FCC9 C3A6BD  Jp    BDA6

```

BASIC 1.0

```

*****
FCCC CD15FE call FE15      opérateur Basic '+'
FCCF 3009  jr  nc,FCDA    tester type des opérandes
FCD1 CDACBD call BDAC     virgule flottante ?
FCD4 DA0DFF jp          c,FF0D addition entiers hl := hl + de
                                pas de dépassement, accepter
                                résultat dans hl
FCD7 CD4FFE call FE4F     convertir en virgule flottante
FCDA CD58BD call BD58     addition avec virgule flottante
FCDD D8    ret  c        pas de dépassement, ok
FCDE C3F3CA jp          CAF3 'Overflow'

*****
FCE1 CD15FE call FE15     BASIC-Operator '-'
FCE4 3009  jr  nc,FCEF    tester type des opérandes
FCE6 CDB2BD call BDB2     virgule flottante ?
FCE9 DA0DFF jp          c,FF0D soustraction entiers hl := de - hl
                                pas de dépassement, résultat dans hl
FCEC CD4FFE call FE4F     convertir en format virgule flott.
FCEF CD5EBD call BD5E     soustraction virgule flottante
FCF2 D8    ret  c        pas de dépassement, ok
FCF3 18E9  jr  FCDE      'Overflow'

*****
FCF5 CD15FE call FE15     opérateur Basic '*'
                                tester type des opérandes

```

BASIC 1.0

FCF8	3009	jr	nc,FD03	virgule flottante ?
FCFA	CDB5BD	call	BDB5	multiplication entiers avec signe
FCFD	DAODFF	jp	c,FF0D	pas de dépassement, accepter
				résultat dans hl
FD00	CD4FFE	call	FE4F	convertir en virgule flottante
FD03	CD61BD	call	BD61	multiplication à virgule flottante
FD06	D8	ret	c	
FD07	18D5	jr	FCDE	'Overflow'

***** comparaison arithmétique

FD09	CD15FE	call	FE15	tester type des opérands
FD0C	DAC4BD	jp	c,BDC4	comparaison entiers
FD0F	C36ABD	jp	BD6A	comparaison virgule flottante

***** opérateur Basic '/'

FD12	3AC1B0	ld	a,(BOC1)	type de variable
FD15	B1	or	c	
FD16	FE02	cp	02	
FD18	2005	jr	nz,FD1F	
FD1A	CD4FFE	call	FE4F	opérands entiers en virgule flottante

FD1D	1803	jr	FD22	
FD1F	CD15FE	call	FE15	tester type des opérands
FD22	EB	ex	de,hl	
FD23	D5	push	de	
FD24	CD64BD	call	BD64	division virgule flottante
FD27	D1	pop	de	
FD28	F5	push	af	
FD29	010500	ld	bc,0005	
FD2C	CDF2FF	call	FFF2	ldir
FD2F	F1	pop	af	
FD30	D8	ret	c	ok ?
ED31	CAEACA	jp	z,CAEA	'Division by zero'
FD34	C3F3CA	jp	CAF3	'Overflow'

***** opérateur Basic 'Backslash'

FD37	CD9AFE	call	FE9A	
FD3A	EB	ex	de,hl	
FD3B	CDB8BD	call	BDB8	division entiers avec signe

BASIC 1.0

FD3E	DA0DFF	jp	c,FF0D	accepter résultat dans hl
FD41	2810	jr	z,FD53	'Division by zero'
FD43	210080	ld	hl,8000	
FD46	C360FE	jp	FE60	

***** opérateur Basic 'MOD'

FD49	CD9AFE	call	FE9A	
FD4C	EB	ex	de,hl	
FD4D	CDBBBD	call	BDBB	calcul MOD
FD50	DA0DFF	jp	c,FF0D	accepter résultat dans hl
FD53	1E0B	ld	e,0B	'Division by zero'
FD55	C394CA	jp	CA94	sortir message d'erreur

***** opérateur Basic 'AND'

FD58	CD9AFE	call	FE9A	
FD5B	7B	ld	a,e	
FD5C	A5	and	l	
FD5D	6F	ld	l,a	hl and de
FD5E	7C	ld	a,h	
FD5F	A2	and	d	
FD60	C30CFF	jp	FF0C	

***** opérateur Basic 'OR'

FD63	CD9AFE	call	FE9A	
FD66	7B	ld	a,e	
FD67	B5	or	l	
FD68	6F	ld	l,a	hl or de
FD69	7A	ld	a,d	
FD6A	B4	or	h	
FD6B	18F3	jr	FD60	

***** opérateur Basic 'XOR'

FD6D	CD9AFE	call	FE9A	
FD70	7B	ld	a,e	
FD71	AD	xor	l	
FD72	6F	ld	l,a	hl xor de
FD73	7C	ld	a,h	
FD74	AA	xor	d	
FD75	18E9	jr	FD60	

BASIC 1.0

```

*****
FD77 E5      push   hl          opérateur Basic NOT
FD78 CD8DFE  call   FE8D          CINT
FD7B 7D      ld      a,l
FD7C 2F      cpl    a          compléter Lo-Byte
FD7D 6F      ld      l,a
FD7E 7C      ld      a,h
FD7F 2F      cpl    a
FD80 CDOCFE  call   FFOC          compléter Hi-Byte
FD83 E1      pop    hl
FD84 C9      ret

*****
FD85 CDA3FD  call   FDA3          fonction Basic ABS
FD88 F0      ret      p          SGN
*****
                               signe positif, terminé
                               inverser signe
FD89 E5      push   hl
FD8A C5      push   bc
FD8B CD2DFF  call   FF2D          amener résultat numérique
FD8E 300D    jr     nc,FD9D      inversion de signe virgule flottante
FD90 CDC7BD  call   BDC7          inversion de signe entier
FD93 22C2B0  ld     (B0C2),hl
FD96 D5      push   de
FD97 D460FE  call   nc,FE60
FD9A D1      pop    de
FD9B 1803    jr     FDA0

*****
                               inversion de signe virgule flottante
                               inversion de signe virgule flottante
FD9D CD6DBD  call   BD6D
FDA0 C1      pop    bc
FDA1 E1      pop    hl
FDA2 C9      ret

*****
FDA3 CD2DFF  call   FF2D          amener résultat numérique
FDA6 DACABD  jp     c,BDCA        SGN entier
FDA9 C5      push   bc
FDA A CD70BD  call   BD70          SGN virgule flottante
FDAD C1      pop    bc
FDAE C9      ret

```

BASIC 1.0

```

*****
FDAF E5      push  hl
FDB0 79      ld     a,c
FDB1 CD4BFF  call  FF4B      accepter type et valeur de variable
FDB4 D1      pop   de
FDB5 CD2DFE  call  FF2D      aller chercher résultat numérique
FDB8 78      ld     a,b      chiffres d'arrondissement
FDB9 300B    jr     nc,FDC6   valeur à virgule flottante ?
FDBB B7      or     a
FDBC F0      ret    p      arrondi après la virgule? terminé
FDBD CD6AFE  call  FE6A      convertir valeur entière en virgule
                                flottante
FDC0 CDCEFD  call  FDCE~     arrondir nombre
FDC3 C38DFE  jp     FE8D     CINT

*****
FDC6 B7      or     a      arrondir nombre à virgule flottante
                                chiffres d'arrondissement
FDC7 2005    jr     nz,FDCE  différent zéro, alors arrondir
FDC9 1149BD  ld     de,BD49   convertir virgule flottante en
                                entier
FDCC 1826    jr     FDF4

*****
FDCE D5      push  de
FDCF C5      push  bc
FDD0 78      ld     a,b      chiffres d'arrondissement
FDD1 CD55BD  call  BD55      multiplier nombre à virgule
                                flottante par 10^a
FDD4 DC49BD  call  c,BD49    convertir virgule flottante en
                                entier
FDD7 78      ld     a,b
FDD8 C1      pop   bc
FDD9 D1      pop   de
FDDA 3008    jr     nc,FDE4
FDDC CD43BD  call  BD43      convertir entier en virgule
                                flottante
FDDF AF      xor   a      inverser chiffres d'arrondissement
FDE0 90      sub   b      correspond division
FDE1 C355BD  jp     BD55      multiplier nombre à virgule
                                flottante par 10^a

```


BASIC 1.0

FDE4	EB	ex	de,h1	
FDE5	C34EFF	jp	FF4E	
*****				fonction Basic FIX
FDE8	114CBD	ld	de,BD4C	fonction FIX
FDEB	1803	Jr	FDF0	
*****				fonction Basic INT
FDED	114FBD	ld	de,BD4F	fonction INT
FDF0	CD2DFF	call	FF2D	amener résultat numérique
FDF3	D8	ret	c	entier ?
FDF4	CDFBFF	call	FFF8	jp (de)
FDF7	D0	ret	nc	
FDF8	3AC1B0	ld	a,(BOC1)	type de variable
FDFB	CD06FE	call	FE06	
FDFE	D8	ret	c	
FDFE	CD1DFF	call	FF1D	type de variable dans c, pointeur dans h1
FE02	78	ld	a,b	
FE03	C343BD	jp	BD43	convertir entier en virgule flottante
FE06	79	ld	a,c	
FE07	FE03	cp	03	'chaîne' ?
FE09	D0	ret	nc	
FE0A	7E	ld	a,(h1)	
FE0B	23	inc	h1	
FE0C	66	ld	h,(h1)	
FE0D	6F	ld	l,a	
FE0E	CDA9BD	call	BDA9	si positif, accepter signe de b
FE11	D0	ret	nc	
FE12	C30DFF	jp	FF0D	accepter résultat dans h1
FE15	79	ld	a,c	
FE16	FE03	cp	03	'chaîne' ?
FE18	2832	Jr	z,FE4C	oui, 'Type mismatch'
FE1A	3AC1B0	ld	a,(BOC1)	type de variable
FE1D	FE03	cp	03	'chaîne'
FE1F	282B	Jr	z,FE4C	oui, 'Type mismatch'
FE21	B9	cp	c	

BASIC 1.0

FE22	2817	Jr	z,FE3B	
FE24	300C	Jr	nc,FE32	
FE26	E5	push	hl	
FE27	21C1B0	ld	hl,BOC1	type de variable
FE2A	71	ld	(hl),c	
FE2B	23	inc	hl	
FE2C	CD63FE	call	FE63	convertir nombre entier en virgule flottante
FE2F	D1	pop	de	
FE30	B7	or	a	
FE31	C9	ret		
FE32	CD63FE	call	FE63	convertir nombre entier en virgule flottante
FE35	EB	ex	de,hl	
FE36	21C2B0	ld	hl,BOC2	
FE39	B7	or	a	
FE3A	C9	ret		
FE3B	EE02	xor	02	
FE3D	2805	Jr	z,FE44	
FE3F	EB	ex	de,hl	
FE40	21C2B0	ld	hl,BOC2	
FE43	C9	ret		
FE44	5E	ld	e,(hl)	
FE45	23	inc	hl	
FE46	56	ld	d,(hl)	
FE47	2AC2B0	ld	hl,(BOC2)	
FE4A	37	scf		
FE4B	C9	ret		
FE4C	C340FF	jp	FF40	'Type mismatch'
*****				opérandes entiers en virgule flottante
FE4F	2AC2B0	ld	hl,(BOC2)	premier opérande
FE52	CD6AFE	call	FE6A	convertir
FE55	2A8BB0	ld	hl,(B08B)	pointeur de pile Basio, second opérande
FE58	CD63FE	call	FE63	convertir

BASIC 1.0

```

FE5B EB      ex      de,h1
FE5C 21C2B0  ld      hl,BOC2
FE5F C9      ret

FE60 AF      xor      a
FE61 1808    jr      FE6B      convertir en virgule flottante

***** convertir nombre entier en virgule flottante
FE63 5E      ld      e,(hl)
FE64 23      inc     hl
FE65 56      ld      d,(hl)
FE66 2B      dec     hl
FE67 7A      ld      a,d
FE68 1808    jr      FE72

***** convertir nombre entier en virgule flottante
FE6A 7C      ld      a,h
FE6B EB      ex      de,h1
FE6C 21C1B0  ld      hl,BOC1      type de variable
FE6F 3605    ld      (hl),05      'Real'
FE71 23      inc     hl
FE72 EB      ex      de,h1
FE73 F5      push   af
FE74 B7      or      a
FE75 FCC7BD  call   m,BDC7      si négatif, inversion de signe
                          entier
FE78 F1      pop     af
FE79 C340BD  jp     BD40      convertir nombre entier en virgule
                          flottante

***** convertir valeur 4 octets en virgule flottante
FE7C 22C2B0  ld      (BOC2),hl    Lo-Word
FE7F EB      ex      de,h1
FE80 22C4B0  ld      (BOC4),hl    Hi-Word
FE83 21C1B0  ld      hl,BOC1      type de variable
FE86 3605    ld      (hl),05      'Real'
FE88 23      inc     hl      pointeur sur valeur 4 octets
FE89 AF      xor      a
FE8A C343BD  jp     BD43      convertir en virgule flottante

```

BASIC 1.0

```

*****
fonction Basic CINT
FE8D CD93FE call FE93
FE90 D8 ret c
FE91 183F Jr FED2 'Overflow'

FE93 CDA5FE call FEA5
FE96 22C2B0 ld (BOC2),hl
FE99 C9 ret

FE9A 79 ld a,c
FE9B CDACFE call FEAC
FE9E EB ex de,hl
FE9F DCA5FE call c,FEA5
FEA2 D8 ret c
FEA3 182D Jr FED2 'Overflow'

FEA5 21C1B0 ld hl,BOC1 type de variable
FEA8 7E ld a,(hl)
FEA9 3602 ld (hl),02 'Integer'
FEAB 23 inc hl
FEAC FE03 cp 03 'chafne' ?
FEAE 380D Jr c,FEBD
FEB0 CA40FF Jp z,FF40 'Type mismatch'
FEB3 C5 push bc
FEB4 CD46BD call BD46 convertir nombre à virgule flottante
en entier

FEB7 47 ld b,a
FEB8 DCA9BD call c,BDA9 accepter signe b dans nombre entier
hl

FEBB C1 pop bc
FEBC C9 ret

*****
valeur entière (hl) dans hl
FEBD 7E ld a,(hl)
FEBE 23 inc hl
FEBF 66 ld h,(hl)
FECO 6F ld l,a
FEC1 C9 ret

*****
fonction Basic UNT

```

BASIC 1.0

FEC2	CD2DFF	call	FF2D	amener résultat numérique
FEC5	D8	ret	c	entier ?
FEC6	CD46BD	call	BD46	convertir nombre virgule flottante en entier
FEC9	3007	jr	nc,FED2	'Overflow'
FECB	47	ld	b,a	
FEC C	FCA9BD	call	m,BDA9	accepter signe b dans nombre entier h1
FECF	DA0DFF	jp	c,FF0D	accepter nombre entier dans h1
FED2	1E06	ld	e,06	'Overflow'
FED4	C394CA	jp	CA94	sortir message d'erreur
FED7	E5	push	h1	
FED8	D5	push	de	
FED9	C5	push	bc	
FEDA	21C1B0	ld	h1,BOC1	type de variable
FEDD	BE	cp	(h1)	
FEDE	C4E5FE	call	nz,FEE5	
FEE1	C1	pop	bc	
FEE2	D1	pop	de	
FEE3	E1	pop	h1	
FEE4	C9	ret		
FEE5	D603	sub	03	
FEE7	38A4	jr	c,FE8D	CINT
FEE9	CA3CFF	jp	z,FF3C	type 'chaîne', sinon 'Type mismatch'
*****				fonction Basic CREAL
FEEC	CD2DFF	call	FF2D	amener résultat numérique
FE EF	DA6AFE	jp	c,FE6A	entier, alors convertir
FEF2	C9	ret		
*****				fixer valeur virgule flottante sur zéro
FEF3	E5	push	h1	
FEF4	210000	ld	h1,0000	
FEF7	22C2B0	ld	(BOC2),h1	
FEFA	22C4B0	ld	(BOC4),h1	
FEFD	22C5B0	ld	(BOC5),h1	
FF00	E1	pop	h1	
FF01	C9	ret		

BASIC 1.0

```

*****          fonction Basic  SGN
FF02  CDA3FD    call  FDA3      SGN
FF05  6F        ld     l,a
FF06  87        add   a,a
FF07  9F        sbc   a,a
FF08  1802     jr     FFOC

*****          accepter contenu accu comme nombre entier
FF0A  6F        ld     l,a      Lo-Byte
FF0B  AF        xor   a        annuler Hi-Byte
FF0C  67        ld     h,a

*****          accepter nombre entier dans hl
FF0D  22C2B0   ld     (BOC2),hl  nombre dans BOC2
FF10  3E02     ld     a,02      type sur 'entier'
FF12  32C1B0   ld     (BOC1),a   type de variable
FF15  C9        ret

*****          type de variable sur virgule
                    flottante
FF16  21C2B0   ld     hl,BOC2   pointeur sur nombre à virgule
                    flottante
FF19  3E05     ld     a,05      type sur 'Real'
FF1B  18F5     jr     FF12

*****          amener type de variable, hl est pointé sur variable
FF1D  21C1B0   ld     hl,BOC1   type de variable
FF20  4E        ld     c,(hl)   dans c
FF21  23        inc   hl        hl pointé sur variable
FF22  C9        ret

*****          amener type de variable
FF23  3AC1B0   ld     a,(BOC1)   type de variable
FF26  C9        ret

*****          tester si chaîne
FF27  3AC1B0   ld     a,(BOC1)  type de variable
FF2A  FE03     cp     03        'chaîne' ?
FF2C  C9        ret

```

BASIC 1.0

```

*****
FF2D 3AC1B0 ld a,(BOC1) amener résultat numérique
FF30 FE03 cp 03 type de variable
FF32 280C jr z,FF40 chaîne ?
FF34 2AC2B0 ld hl,(BOC2) oui, 'Type mismatch'
FF37 D8 ret c charger valeur entière
FF38 21C2B0 ld hl,BOC2 pas virgule flottante, terminé
                                adresse du nombre à virgule
                                flottante
FF3B C9 ret

*****
FF3C CD45FF call FF45 tester si chaîne
FF3F C8 ret z oui, ok
FF40 1E0D ld e,0D 'Type mismatch'
FF42 C394CA jp CA94 sortir message d'erreur

*****
FF45 3AC1B0 ld a,(BOC1) tester si chaîne
FF48 FE03 cp 03 type de variable
FF4A C9 ret 'chaîne' ?

*****
FF4B 32C1B0 ld (BOC1),a type de variable
FF4E 11C2B0 ld de,BOC2
FF51 1813 jr FF66

*****
FF53 D5 push de placer résultat sur pile Basic
FF54 E5 push hl
FF55 3AC1B0 ld a,(BOC1) type de variable
FF58 4F ld c,a
FF59 CDB0F5 call F5B0 réserver place dans pile Basic
FF5C CD62FF call FF62 placer sur pile
FF5F E1 pop hl
FF60 D1 pop de
FF61 C9 ret

*****
FF62 EB ex de,hl copier variable dans (hl)
FF63 21C2B0 ld hl,BOC2

```

BASIC 1.0

FF66	C5	push	bc	
FF67	3AC1B0	ld	a,(B0C1)	type de variable
FF6A	4F	ld	c,a	
FF6B	0600	ld	b,00	
FF6D	EDB0	ldir		copier résultat
FF6F	C1	pop	bc	
FF70	C9	ret		

*****				tester si lettre
FF71	CD8AFF	call	FF8A	convertir minuscules en majuscules
FF74	FE41	cp	41	'A'
FF76	3F	ccf		
FF77	D0	ret	nc	
FF78	FE5B	cp	5B	'Z'+1
FF7A	C9	ret		

*****				tester si caractères alphanumériques
FF7B	CD71FF	call	FF71	tester si lettres
FF7E	D8	ret	c	oui
FF7F	FE2E	cp	2E	','
FF81	37	scf		
FF82	C8	ret	z	
FF83	FE30	cp	30	'0'
FF85	3F	ccf		
FF86	D0	ret	nc	
FF87	FE3A	cp	3A	'9'+1
FF89	C9	ret		

*****				conversion minuscules-majuscules
FF8A	FE61	cp	61	'a'
FF8C	D8	ret	c	
FF8D	FE7B	cp	7B	'z'+1
FF8F	D0	ret	nc	
FF90	D620	sub	20	'a'-'A'
FF92	C9	ret		

*****				parcourir table suivante
FF93	F5	push	af	
FF94	C5	push	bc	
FF95	46	ld	b,(h1)	charger longueur de table

BASIC 1.0

FF96	23	inc	hl	
FF97	E5	push	hl	adresse de retour si rechercher négative
FF98	23	inc	hl	pointeur sur valeur suivante table
FF99	23	inc	hl	
FF9A	BE	cp	(hl)	comparer caractère
FF9B	23	inc	hl	augmenter pointeur
FF9C	2804		jr	z,FFA2 trouvé
FF9E	05	dec	b	diminuer compteur
FF9F	20F7	jr	nz,FF98	table pas encore finie ?
FFA1	E3	ex	(sp),hl	charger adresse de retour
FFA2	F1	pop	af	
FFA3	7E	ld	a,(hl)	
FFA4	23	inc	hl	
FFA5	66	ld	h,(hl)	adresse dans hl
FFA6	6F	ld	l,a	
FFA7	C1	pop	bc	
FFA8	F1	pop	af	
FFA9	C9	ret		
*****				parcourir zone de mémoire (hl)
FFAA	C5	push	bc	Jusqu'à (hl) = a (c=1) ou (hl) = 0 (c=0)
FFAB	4F	ld	c,a	a dans c
FFAC	7E	ld	a,(hl)	
FFAD	B7	or	a	
FFAE	2805	jr	z,FFB5	zéro ?
FFB0	23	inc	hl	
FFB1	B9	cp	c	
FFB2	20F8	jr	nz,FFAC	égale a originaire a ?
FFB4	37	scf		mettre carry
FFB5	79	ld	a,c	
FFB6	C1	pop	bc	
FFB7	C9	ret		
*****				test hl = de ?
FFB8	7C	ld	a,h	
FFB9	92	sub	d	h - d
FFBA	C0	ret	nz	
FFBB	7D	ld	a,l	

BASIC 1.0

```

FFBC 93      sub    e          l - e
FFBD C9      ret

*****      test  hl = bc ?
FFBE 7C      ld     a,h
FFBF 90      sub    b          h - b
FFC0 C0      ret    nz
FFC1 7D      ld     a,l        l - c
FFC2 91      sub    c
FFC3 C9      ret

*****      de := de - hl
FFC4 C5      push   bc          sauver bc
FFC5 47      ld     b,a          sauver a
FFC6 7D      ld     a,l
FFC7 93      sub    e          e - l
FFC8 5F      ld     e,a
FFC9 7C      ld     a,h
FFCA 9A      sbc   a,d          d - h
FFCB 57      ld     d,a
FFCC 78      ld     a,b          ramener a
FFCD C1      pop    bc          ramener bc
FFCE C9      ret

*****      hl := hl - de
FFCF C5      push   bc          sauver bc
FFD0 47      ld     b,a          sauver a
FFD1 7D      ld     a,l
FFD2 93      sub    e          l - e
FFD3 6F      ld     l,a
FFD4 7C      ld     a,h
FFD5 9A      sbc   a,d          h - d
FFD6 67      ld     h,a
FFD7 78      ld     a,b          ramener a
FFD8 C1      pop    bc          ramener bc
FFD9 C9      ret

*****      bc := hl - de
FFDA E5      push   hl          sauver hl
FFDB 67      ld     h,a          sauver a

```

BASIC 1.0

FFDC	E3	ex	(sp),hl	rétablir hl
FFDD	7D	ld	a,l	
FFDE	93	sub	e	l - e
FFDF	4F	ld	c,a	dans c
FFE0	7C	ld	a,h	
FFE1	9A	sbc	a,d	h - d
FFE2	47	ld	b,a	dans b
FFE3	E3	ex	(sp),hl	
FFE4	7C	ld	a,h	ramener a
FFE5	E1	pop	hl	ramener hl
FFE6	C9	ret		

***** hl := hl - bc

FFF7	D5	push	de	sauver de
FFE8	57	ld	d,a	sauver a
FFE9	7D	ld	a,l	
FFEA	91	sub	c	l - c
FFEB	6F	ld	l,a	
FFEC	7C	ld	a,h	
FFED	98	sbc	a,b	h - b
FFEE	67	ld	h,a	
FFEF	7A	ld	a,d	ramener a
FFF0	D1	pop	de	ramener de
FFF1	C9	ret		

***** transfert de bloc ldir

FFF2	EDB0	ldir	
FFF4	C9	ret	

***** transfert de bloc lddr

FFF5	EDB8	lddr	
FFF7	C9	ret	

***** saut dans (hl)

FFF8	E9	jp	(hl)
------	----	----	------

***** saut dans (bc)

FFF9	C5	push	bc
FFFA	C9	ret	

BASIC 1.0

```
*****          saut dans (de)
FFFB D5      push  de
FFFC C9      ret

FFFD C7      rst   0
FFFE C7      rst   0
FFFF 54
```

4 ANNEXES

4.1 Les routines du système d'exploitation

Nous avons établi ici une liste des routines et des tableaux du système d'exploitation, pour autant qu'elles nous soient connues.

Attention! N'essayez jamais d'appeler ces routines avec les adresses que nous vous fournissons ici, si vous ne maîtrisez pas pleinement le mécanisme de commutation de la configuration de la mémoire!

Utilisez plutôt les vecteurs qui vous sont fournis au chapitre 2.1.

Cette présentation a pour but principal de vous permettre de retrouver aisément dans le listing les vecteurs portant les mêmes noms.

0030 RST 6 USER
0040 Jusqu'ici, recopié dans la Ram
0044 Restore High Kernel Jumps
005C KL CHOKE OFF
0099 KL TIME PLEASE
00A3 KL TIME SET
00B1 Scan Events
0153 Kick Event
0163 KL NEW FRAME FLY
016A KL ADD FRAME FLY
0176 KL NEW FAST TICKER
017D KL ADD FAST TICKER
0183 Delete Fast Ticker
0189 Traiter Ticker Chain
01B3 KL ADD TICKER
01C5 Delete Ticker
01D2 KL INIT EVENT
01E2 KL EVENT
021A KL DO SYNC
0228 KL SYNC RESET
022F Ajouter Sync Event
0256 KL NEXT SYNC
0277 KL DONE SYNC
0285 KL DEL SYNCHRONOUS
028E KL DISARM EVENT
0295 KL EVENT DISABLE
029B KL EVENT ENABLE
02A1 KL LOG EXT
02B2 KL FIND COMMAND

0329 KL ROM WALK
0332 KL INIT BACK
0373 Add Event
0382 Delete Event
03B2 KL POLL SYNCHRONOUS
03CA RST 7 INTERRUPT ENTRY CONT'D
0401 EXT INTERRUPT ENTRY
040D KL LOW PCHL CONT'D
0413 RST 1 LOW JUMP CONT'D
0442 KL FAR PCHL CONT'D
044A KL FAR ICALL CONT'D
0450 RST 3 LOW FAR CALL CONT'D
04A1 KL SIDE PCHL CONT'D
04A7 RST 2 LOW SIDE CALL CONT'D
04BF RST 5 FIRM JUMP CONT'D
04DB KL L ROM ENABLE
04E5 KL L ROM DISABLE
04EF KL U ROM ENABLE
04F9 KL U ROM DISABLE
0503 KL ROM RESTORE
050F KL ROM SELECT
0514 KL PROBE ROM
051D KL ROM DESELECT
0533 KL CURR SELECTION
0537 KL LDIR
053D KL LDDR
0543 Rom off & config. save
055C RAM LAM
056D RAM LAM (IX)
0580 RESET CONT'D
05B4 Table 60Hz
05C4 Table 50Hz
05DC MC BOOT PROGRAM
060B MC START PROGRAM
065C Démarrage à froid
066D Message après allumage
0693 Message de copyright
06EB Sortir messages
06F4 Message d'erreur de chargement
0727 Noms de sociétés
0776 MC SET MODE
0786 MC CLEAR INKS

0799 MC SET INKS
07AB Sortir couleur
07BA MC WAIT FLYBACK
07C6 MC SCREEN OFFSET
07E6 MC RESET PRINTER
07F2 MC PRINT CHAR
07F8 MC WAIT PRINTER
0807 MC SEND PRINTER
081B MC BUSY PRINTER
0826 MC SOUND REGISTER
0846 Scan Keyboard
0888 JUMP RESTORE
08AC Main Jump Adr.
0A28 Basic Jump Adr.
0A8A Move (h1+3)((h1+1)),cnt=(h1)
0AA0 SCR INITIALISE
0AB1 SCR RESET
0ACA SCR SET MODE
0AEC SCR GET MODE
0AF7 SCR MODE CLEAR
0B11 Charger masques bits
0B2E Bit Masks Mode 2
0B36 Bit Masks Mode 1
0B3A Bit Masks Mode 0
0B3C SCR SET OFFSET
0B45 SCR SET BASE
0B50 SCR GET LOCATION
0B57 SCR CHAR LIMITS
0B64 SCR CHAR POSITION
0BA9 SCR DOT POSITION
0BF9 SCR NEXT BYTE
0C05 SCR PREV BYTE
0C13 SCR NEXT LINE
0C2D SCR PREV LINE
0C49 SCR ACCESS
0C68 SCR WRITE
0C6B SCR PIXELS (FORCE Mode)
0C72 XOR Mode
0C77 AND Mode
0C7D OR Mode
0C82 SCR READ
0C86 SCR INK ENCODE

OCA0 SCR INK DECODE
OCD2 Reset couleurs
OCE4 SCR SET FLASHING
OCE8 SCR GET FLASHING
OCEC SCR SET INK
OCF1 SCR SET BORDER
OCF2 Set Colour
ODOA Aller chercher entrée matrice de couleur
OD14 SCR GET INK
OD19 SCR GET BORDER
OD1A Get Colour
OD2F Aller chercher adr. ink
OD5B Set Inks on Frame Fly
OD6D Flash Inks
OD81 Aller chercher params de jeu de couleurs actuel
OD93 Matrice de couleur
ODB3 SCR FILL BOX
ODB7 SCR FLOOD BOX
ODDF SCR CHAR INVERT
ODF2 Adresser la mémoire couleur
ODFA SCR HW ROLL
OE3E SCR SW ROLL
OEF3 SCR UNPACK
OF49 SCR REPACK
OFC4 SCR HORIZONTAL
102F SCR VERTICAL
104D Couleur par défaut
1078 TXT INITIALISE
1088 TXT RESET
10A3 Reset Params (toutes les fenêtres)
10E8 TXT STR SELECT
1107 TXT SWAP STREAMS
1122 ldir cnt=15
112A Adr. params fenêtres => de
113D Fixer params TXT par défaut
115E TXT SET COLUMN
1169 TXT SET ROW
1174 TXT SET CURSOR
1180 TXT GET CURSOR
118A Fenêtre act. haut,gauche+hl
1197 Fenêtre act. haut,gauche-hl
11A8 move Cursor

11CE TXT VALIDATE
11DA hl dans limites fenetre?
120C TXT WIN ENABLE
1256 TXT GET WINDOW
1263 TXT DRAW/UNDRAW CURSOR
1268 TXT PLACE/REMOVE CURSOR
1279 TXT CUR ON
1281 TXT CUR OFF
1289 TXT CUR ENABLE
128B Cur Enable Cont'd
129A TXT CUR DISABLE
129C Cur Disable Cont'd
12A9 TXT SET PEN
12AE TXT SET PAPER
12BD TXT GET PEN
12C3 TXT GET PAPER
12C9 TXT INVERSE
12D3 TXT GET MATRIX
12F1 TXT SET MATRIX
12FD TXT SET M TABLE
132A TXT GET M TABLE
1334 TXT WR CHAR
134A TXT WRITE CHAR
137A TXT SET BACK
1387 TXT GET BACK
13A7 TXT SET GRAPHIC
13AB TXT RD CHAR
13C0 TXT UNWRITE
1400 TXT OUTPUT
140C TXT OUT ACTION
144B TXT VDU DISABLE
1451 TXT VDU ENABLE
146B Sauts caractères de commande par défaut
14CB TXT GET CONTROLS
14D8 07 Bip
14E3 16 Transparentmode mis/éteint
14E8 1C =INK (instruction)
14F1 1D =BORDER (instruction)
14F8 1A définir fenetre
1504 19 =SYMBOL (instruction)
150A 08 CRSR LEFT
150F 09 CRSR RGHT

1514 OA CRSR DOWN
1519 OB CRSR UP
152A 1E CRSR HOME
1530 OD CRSR sur début de ligne
1538 1F =LOCATE (instruction)
1540 TXT CLEAR WINDOW
154F 10 supprimer caractère sur CRS Pos
1556 14 vider fenêtre à partir de CRS Pos
156D 13 vider fenêtre Jusqu'à CRS Pos
1584 12 supprimer ligne à partir de CRS Pos
158E 11 supprimer ligne Jusqu'à CRS Pos
15B0 GRA INITIALISE
15DF GRA RESET
15F1 GRA MOVE RELATIVE
15FC GRA ASK CURSOR
1612 GRA GET ORIGIN
161A aller chercher position de départ physique
161D aller chercher position objet et fixer Cur
1657 Add coord. act. + coord. rel.
1734 GRA WIN WIDTH
1779 GRA WIN HEIGHT
17A6 GRA GET W WIDTH
17BC GRA GET W HEIGHT
17C5 GRA CLEAR WINDOW
17F6 GRA SET PEN
17FD GRA SET PAPER
1804 GRA GET PEN
180A GRA GET PAPER
1810 GRA PLOT RELATIVE
1813 GRA PLOT ABSOLUTE
1816 GRA PLOT
1824 GRA TEST RELATIVE
1827 GRA TEST ABSOLUTE
182A GRA TEST
1836 GRA LINE RELATIVE
1839 GRA LINE ABSOLUTE
183C GRA LINE
1945 GRA WR CHAR
19E0 KM INITIALISE
1A1E KM RESET
1A3C KM WAIT CHAR
1A42 KM READ CHAR

1A81 KM EXP BUFFER CONT'D
1AB3 Default Exp String
1ABD KM SET EXPAND
1AE5 nettoyer Exp Buffer
1B22 Place pour nouvelle Exp String?
1B2E KM GET EXPAND
1B3E Adr. Exp String dans de
1B56 KM WAIT KEY
1B5C KM READ KEY
1BB3 KM GET STATE
1BB7 Update Key State Map
1C2F KM TEST BREAK
1C5C KM GET JOYSTICK
1C69 KM GET DELAY
1C6D KM SET DELAY
1C71 KM ARM BREAK
1C82 KM DISARM BREAK
1C90 KM BREAK EVENT
1CBD KM TEST KEY
1CCD aller chercher bit correspondant à touche #
1CE5 Bit Masks
1D3E KM GET TRANSLATE
1D43 KM GET SHIFT
1D48 KM GET CONTROL
1D4B Get Key Table
1D52 KM SET TRANSLATE
1D57 KM SET SHIFT
1D5C KM SET CONTROL
1D5F Set Key Table
1D69 Key Translation Table
1DB9 Key SHIFT Table
1E09 Key CTRL Table
1E68 SOUND RESET
1ECB SOUND HOLD
1EE6 SOUND CONTINUE
1F03 Sound Event
1F61 Scan Sound Queues
1F9F SOUND QUEUE
204A SOUND RELEASE
206C SOUND CHECK
2089 SOUND ARM EVENT
2273 fixer volume

2338 SOUND AMPL ENVELOPE
233D SOUND TONE ENVELOPE
2340 copier courbe d'enveloppe
2349 SOUND A ADDRESS
234E SOUND T ADDRESS
2351 aller chercher adresse courbe d'enveloppe
2370 CAS INITIALISE
237F CAS SET SPEED
238E CAS NOISY
2392 CAS IN OPEN
23AB CAS OUT OPEN
23AF CAS Open
23FC CAS IN CLOSE
2401 CAS IN ABANDON
2415 CAS OUT CLOSE
242E CAS OUT ABANDON
2435 CAS IN CHAR
245B CAS OUT CHAR
248B Check Input Buffer Status
248E Check Buffer Status
2496 CAS TEST EOF
249A CAS RETURN
24AB CAS IN DIRECT
24EA CAS OUT DIRECT
2528 CAS CATALOG
253F lire File Header
271F sortir message CAS (# dans b)
2780 sortir message CAS (1 caractère)
27C5 messages cassette
2836 CAS READ
283F CAS WRITE
2851 CAS CHECK
2873 allumer moteur et ouvrir clavier
29CD CAS Input RD DATA & Test ESC
2A37 CAS Output WR DATA
2A4B CAS START MOTOR
2A4F CAS STOP MOTOR
2A51 CAS RESTORE MOTOR
2A98 EDIT
2AC6 EDIT exécuter saut
2AE0 EDIT Table de saut 1
2B1C EDIT Table de saut 2

2B2B BIP
2B2F CRSR UP
2B33 CRSR DWN
2B37 CRSR RGHT
2B3B CRSR LEFT
2B42 ESC
2B61 message *BREAK*
2B69 ENTER
2B75 CRSR RGHT (Buffer)
2B7E CRSR DWN (Buffer)
2B89 CTRL & CRSR RGHT
2B92 CTRL & CRSR DWN
2BAA CRSR LEFT (Buffer)
2BB3 CRSR UP (Buffer)
2BBD CTRL & CRSR LEFT
2BC7 CTRL & CRSR UP
2BF9 CTRL & TAB (Flip Insert)
2C01 insérer caractère
2C3D DEL
2C4A CLR
2C98 SHFT & CRSR RGHT
2C9D SHFT & CRSR LEFT
2CA2 SHFT & CRSR UP
2CA7 SHFT & CRSR DWN
2CEA COPY
2DD9 caractère du clavier
2DF6 aller chercher adr. de saut EDIT
2E18 FLO copier variable de (de) => (h1)
2E29 FLO Int => Flo
2E55 FLO valeur 4 octets => Flo
2E5E FLO valeur 4 octets * 256 => Flo
2E66 FLO Flo => Int
2E8E FLO Flo => Int
2EA1 FLO FIX
2EAC FLO INT
2EB6 FLO
2F94 FLO RND Init
2FA1 FLO set RND seed
2FB7 FLO RND
2FD1 FLO multiplier nombre par 10^a
2FE6 FLO aller chercher dernière valeur RND
300F FLO LOG10

3014 FLO LOG
3090 FLO EXP
310A FLO SQR
310D FLO élévation à la puissance
31A3 FLO PI
31AE FLO DEG/RAD
31B2 FLO COS
31BC FLO SIN
3231 FLO TAN
3241 FLO ATN
3337 FLO soustraction
333B FLO soustraction
333F FLO addition
3415 FLO multiplication
349E FLO division
3578 FLO multiplier le chiffre par 2^a
359A FLO comparer
35E8 FLO SGN
35F8 FLO inverser signe
3708 INT
370E INT
3715 INT accepter signe dans b
3728 INT addition
3730 INT soustraction
3731 INT soustraction
3739 INT multiplication avec signe
3750 INT multiplication sans signe
377A INT division avec signe
3781 INT MOD
378C INT division sans signe
37D4 INT inversion de signe
37E0 INT SGN
37E9 INT comparer

4.2 References à la Ram système

Vous trouverez ci-dessous des références croisées aux endroits où elles sont utilisées pour toutes les adresses de la Ram qui apparaissent dans le listing de la Rom du système d'exploitation.

C'est très utile lorsque vous manipulez les contenus de ces adresses avec vos propres programmes et que vous y trouvez soudain une autre valeur que celle que vous attendiez.

Vous pouvez donc consulter la table suivante qui vous indique quelles routines accèdent à une adresse déterminée.

B100: 0066 00F2 011D 0127 061C
B101: 00EC 061F
B102: 00F5 00FE 0102
B104: 00E2 00F8 0114 0132 0142 03E1
B105: 010A 014E
B187: 009E 00AC 00B1 010E
B189: 009A 00A8
B18B: 00A5
B18C: 00BF 016A 0170
B18E: 00C7 017D 0183
B190: 00DC 0189 01BF 01C5
B192: 00D2
B193: 0257 026F 0288 03B9
B194: 022B 03B2
B195: 0264 026C 0277 0295 029B 03C3
B196: 0231 02B2 030A
B1A6: 02A2 02A6 02BF
B1A8: 0080 034B 0467 0499 0529 0533
B1A9: 0060 0086 0096
B1AA: 0348
B1AB: 005D 0083 04B9
B1C8: 0AEC 0B28
B1C9: 0B40 0B50 0B84 0BDD 0E24 0E37
B1CA: 0B00
B1CB: 0AA8 0B47 0B53 0B8D 0BE6 0E2C
B1CC: 0C61
B1CD: 0C64
B1CF: 0B20 0BF1 0C8E 0CA2 0F08 0F18 0F32 0F66 0F7D 0FA1 1015
B1D7: 0CE4 0CE8 0D8F
B1D8: 0D88
B1D9: 0CD5 0D8C

B1EA: 0D32 0D81
B1FB: 0CDE 0D76 0D84
B1FC: 0D06 0D7D
B1FD: 0D5B 0D70
B1FE: 0D3C 0D4F
B207: 0FDC 0FFE
B20C: 10B3 10B7 10EA 1107 1110
B20D: 10A5
B285: 10A8 1139 1163 116E 117A 1180 11AB 11B1 133F 13B1
1546 1560 1577
B287: 123E 125D
B288: 116A 118A 1197 11F3 122D 1256 152A 1543 1559 1570
B289: 115F 1190 119F 11E1 11E6 1533 1593
B28A: 11FB 1230 1259 1549 155C
B28B: 11DA 11EE 1573 1588
B28C: 1186 11B6
B28D: 1140 1263 1291 12A2
B28E: 1335 1456
B28F: 10CE 10DE 126E 12A9 12BD 12C9 12CF 1391 139F 13C0
B290: 10C8 11C1 12AE 12C3 13D3 1566 157D 1597
B291: 1376 1383 1387
B293: 13A7 140D
B294: 1320 132A
B295: 107C
B296: 1325 1330
B298: 134E 13C3 13E9
B2B8: 1415 1447 145C
B2B9: 142E 143F
B2C3: 1432 1462 14CB
B328: 1604 1612 1637
B32A: 1608 1616 164E
B32C: 15F4 15FC 1658
B32E: 15F8 1600 165E
B330: 1666 16D0 16DA 16DE 1700 1758 17A6 17E2
B332: 1670 16C7 16E8 16F1 170A 175C 17AA
B333: 1671 16D8 16A4 16BD 1720 178A 17BC 17D9
B334: 1672 16E0 16A4 16BD 1720 178A 17BC 17D9
B336: 1683 168D 1691 16B3 1716 178E 17C0 17D5
B338: 17F9 1804 181D 190A 192F
B339: 17EC 1800 180A 19D8
B33A: 1898 18B3 1911 1936

B340: 18BE 18CD 18E1
B342: 1841 184E 1859 185D 18A2 18A6 18F7 18FD 1927 193A
B344: 1845 1860 1864 1872 18A9 18AF 1903 1915 191A 1920
B346: 18BA 18F1
B43C: 19EF
B446: 1A24
B4DE: 1A4C 1A6D
B4DF: 1AAF 1ADA
B4E0: 1A43 1A77
B4E1: 1A8E 1B44
B4E3: 1A8A 1B05
B4E5: 1AAC 1B00 1B11 1B1C 1B22
B4E6: 1B27
B4E7: 19EC 1B8D 1BA6 1BB3
B4E8: 1B76
B4E9: 1C15 1C69 1C6D
B4EA: 1C4F
B4EB: 1A0F 1BCE 1BFD 1CC5
B4ED: 1BC6 1CBE
B4F1: 1C5C
B4F3: 1C2F
B4F4: 1C62
B4F5: 1BBA
B4FF: 1BB7 1BCB
B501: 1BC0
B509: 1BF1 1C09 1C18
B50A: 1BF6 1C23
B50B: 19E7
B50C: 1C7E 1C84 1C90
B50D: 1C74
B51D: 1E9D 1EEB 1F12 1F48 1FAD 1FD2 2052
B520: 206F
B522: 1F74
B539: 208D
B53C: 1CEE 1CFE 1D26
B53E: 1D0F 1D15
B540: 1COD 1DOB 1D22
B541: 1A01 1D3E 1D52
B543: 19FD 1D43 1D57
B545: 19F9 1D48 1D5C
B547: 19F5 1C02 1CA6 1CAE
B550: 1F05 20B2

B551: 1E6D 1EE6 201F 20F5
B552: 1E6A 1ECB 1F61 2283
B554: 1F5B 1F97
B555: 1E70
B55C: 1E80 2125
B59B: 212D 2150
B5DA: 2135 2148
B60A: 219A 2338 2349
B619: 1E7D 2292
B6FA: 233D 234E
B800: 238E 2695 2760
B801: 269A 2705 279F
B802: 2392 23FC 2401 248B 2528 256E 25A9 27BF
B803: 24CF 2530 257D
B805: 2451 2456 24A2 24A6 2580
B807: 25D6 25E1 25F3
B817: 258A
B818: 253F
B819: 23A6
B81A: 243F 244A 244E 249B 249F 24BC 24D6 259A
B81C: 239E 24B2 24B9 24C1 24D2 256B
B81E: 2594 25CA
B81F: 23A2
B847: 23AB 2415 242E 245F 24ED 2667
B848: 2504 251B 262C
B84A: 247F 2484 262F
B84C: 261E 2636
B85C: 265B
B85D: 241F 264D
B85E: 24F9
B85F: 2469 2478 247C 2507 2514 2644 2658
B861: 2632
B863: 2624 2660
B864: 24FC
B866: 2500
B88C: 254C 25DE 25F6 2692
B89D: 258E
B89F: 2567 2597
B8A3: 25D0
B8A6: 24CA
B8CC: 240C 2673
B8CD: 2873 295D 2973

B8CE: 2956 29B3
B8D0: 2A08 2A1B
B8D1: 238A
B8D2: 2A0C
B8D3: 28B1 2990 29A2 29A6
B8DC: 2C1E 2C35 2C5B 2C67 2DCE
B8DD: 2AA5 2BF9 2BFD 2C04
B8DE: 2C72 2C76 2C83 2C94 2CAC 2CC4 2CD5 2CF0 2CFE
2D11 2D1A 2D36

4.3 Les routines de la Rom Basic

C006 initialisation du Basic
C03F 'BASIC 1.0' LF,LF
C052 instruction Basic EDIT
C064 mode READY
C0CC 'Ready', LF
COD3 supprimer mode AUTO
COD6 fixer mode AUTO
C0DF instruction Basic AUTO
C12B instruction Basic NEW
C132 instruction Basic CLEAR
C13E supprimer programme et variables
C18C supprimer variables
C1D0 aller chercher numéro stream
C1E3 tester si numéro stream
C20A instruction Basic PAPER
C212 instruction Basic PEN
C221 instruction Basic BORDER
C22A instruction Basic INK
C23C aller chercher argument(s) < 32
C24C aller chercher argument < 16
C24F instruction Basic MODE
C25A instruction Basic CLS
C262 fonction Basic VPOS
C276 fonction Basic POS
C290 aller chercher position PRINT act.
C2D2 instruction Basic LOCATE
C2E1 instruction Basic WINDOW
C2FD WINDOW SWAP
C312 aller chercher argument < 8
C319 instruction Basic TAG
C320 instruction Basic TAGOFF
C327 aller chercher 2 valeurs 8 bits non nulles
C337 sortir chaîne sur stream zéro
C341 sortir chaîne
C34E sortir linefeed
C386 initialiser l'écran
C3A8 sortir CR & LF
C3B5 sortir caractère sur imprimante
C3DF aller chercher actuelle position imprimante
C3E3 instruction Basic WIDTH

C417 variable réservée EOF
C424 aller chercher un caractère sur canal d'entrée
C42C attendre un caractère du clavier
C439 lire clavier
C453 autoriser interruption par 'Break'
C45E routine Break-Event
C46F attendre frappe d'une touche après 'ESC'
C48C instruction Basic ORIGIN
C4C6 instruction Basic DRAW
C4CB Instruction Basic DRAWR
C4D0 instruction Basic PLOT
C4D5 instruction Basic PLOTR
C4E9 Fonction Basic TEST
C4EE fonction Basic TESTR
C505 instruction Basic MOVE
C50A instruction Basic MOVER
C51A aller chercher 2 arguments entiers
C529 instruction Basic FOR
C5FB instruction Basic NEXT
C632 chercher boucle FOR-NEXT ouverte
C6C7 instruction Basic IF
C6E8 instruction Basic GOTO
C6ED instruction Basic GOSUB
C70F instruction Basic RETURN
C72E instruction Basic chercher GOSUB sur pile Basic
C747 instruction Basic WHILE
C776 instruction Basic WEND
C7E3 instruction Basic ON
C807 Traitement Event
C8CB instruction Basic ON BREAK
C8E1 instruction Basic DI
C8E7 instruction Basic EI
C8ED Reset SOUND et Event
C940 instruction Basic ON SQ
C95D calculer adresse de la Sound Queue
C971 instruction Basic AFTER
C979 instruction Basic EVERY
C99F instruction Basic REMAIN
C9B1 calculer adresse du bloc Event
C9C5 chercher NEXT correspondant
CA18 chercher WEND correspondant
CA3B aller chercher ligne d'entrée

CA43 éditer ligne
CA4C aller chercher ligne d'entrée dans cassette
CA84 annuler numéro et ligne d'erreur
CA8F instruction Basic ERROR
CA94 sortir message d'erreur
CB23 'Undefined line'
CB4F 'Break', 'in'
CB5A instruction Basic STOP
CB65 instruction Basic END
CBC0 instruction Basic CONT
CBE5 ON ERROR
CBF8 ON ERROR GOTO 0
CC03 instruction Basic RESUME
CC45 fixer pointeur sur message d'erreur
CC5B messages d'erreur
CE67 aller chercher valeur 8 bits
CE6D aller chercher valeur 8 bits non nulle
CE7C aller chercher valeur 16 bits entre 0 et 32767
CE86 aller chercher valeur entière entre -32768 et +32767
CE91 aller chercher valeur 16 bits entre -32768 et +65535
CE9F aller chercher expression chaîne, préparer paramètres
CEA5 aller chercher expression chaîne
CEB0 aller chercher zone de numéros de ligne
CEE1 aller chercher numéro de ligne dans de
CEFB aller chercher expression
CF07 aller chercher terme
CF30 opérateurs arithmétiques
CF59 opérateurs de comparaison
CF81 table des opérateurs Basic
CFA9 comparaison arithmétique
CFB9 signe négatif
CFC2 opérateur Basic NOT
CFCE aller chercher expression
D00D aller chercher variable
D02C aller chercher constante numérique
D070 aller chercher expression entre parenthèses
D080 calcul de fonction
DOCA adresses des variables réservées
DODC variable réservée ERR
DOE5 variable réservée TIME
DOEE variable réservée ERL
DOF4 variable réservée HIMEM

D0FA pointeur de variable, 'arobas'
D107 variable réservée XPOS
D10F variable réservée YPOS
D117 instruction Basic DEF
D130 fonction Basic FN
D190 adresses des fonctions Basic
D1EA fonction Basic MIN
D1EE fonction Basic MAX
D219 fonction Basic ROUND
D246 instruction Basic CAT
D256 instruction Basic OPENOUT
D25F instruction Basic OPENIN
D298 instruction Basic CLOSEIN
D2A1 instruction Basic CLOSEOUT
D2AD interrompre I/O cassette
D2C0 instruction Basic SOUND
D30D aller chercher valeur 8 bits, s'il y en a une
D31E instruction Basic RELEASE
D329 fonction Basic SQ
D341 aller chercher argument entre -128 et +127
D34E instruction Basic ENV
D385 instruction Basic ENT
D3FF aller chercher argument entre 0 et 4095
D409 fonction Basic INKEY
D423 fonction Basic JOY
D439 instruction Basic KEY
D456 KEY DEF
D494 instruction Basic SPEED
D4AB SPEED KEY & INK
D4C3 SPEED WRITE
D4DB variable réservée pi
D4E7 instruction Basic DEG
D4EB instruction Basic RAD
D4EF fonction Basic SQR
D4F4 opérateur Basic '^'
D519 appeler fonctions arithmétiques
D520 fonction Basic EXP
D525 fonction Basic LOG10
D52A fonction Basic LOG
D52F fonction Basic SIN
D534 fonction Basic COS
D539 fonction Basic TAN

D53E fonction Basic ATN
D559 instruction Basic RANDOMIZE
D584 fonction Basic RND
D5AE restaurer pointeur de variable
D5D2 annuler flag pour FN
D5EA calculer adresse de table pour tableau
D5FC types de variable A-Z sur 'Real'
D614 instruction Basic DEFSTR
D618 instruction Basic DEFINT
D61C instruction Basic DEFREAL
D654 instruction Basic LET
D67D instruction Basic DIM
D686 chercher variable Basic
D690 aller chercher adresse de variable
D708 chercher tableau
D7B5 dimensionnement de variable
D7DB tester si variable dimensionnée
D906 aller chercher nom de variable
D97F déterminer type de variable
D999 actualiser table de tableaux
D9C0 instruction Basic ERASE
D9CC supprimer un tableau
DAF8 instruction Basic LINE
DB1A aller chercher entrée sur appareil actif
DB2B instruction Basic INPUT
DB47 aller chercher entrée et convertir
DB77 '?Redo from start'
DBAD aller chercher entrée du clavier
DCD9 instruction Basic RESTORE
DCEB instruction Basic READ
DD37 tester si encore un caractère
DD3F ignorer les espaces
DD4A tester si fin de l'instruction
DD55 tester si prochain caractère est une virgule
DD61 ignorer espace, TAB et LF
DD71 boucle de l'interpréteur
DDAB exécuter instruction Basic
DDD2 aller chercher actuelle adresse de ligne
DDD6 aller chercher actuelle adresse de ligne et tester si mode direct
DDE2 instruction Basic TRON
DDE6 instruction Basic TROFF
DDEB routine TRACE

DE01 adresses des instructions Basic
 DEE1 aller chercher caractère dans buffer d'entrée
 DFDC table des instructions Basic avec numéro de ligne
 E0F7 instruction Basic LIST
 E10D lister lignes Basic bc-de
 E145 sortir un caractère du clavier
 E155 sortie sur écran
 E163 lister ligne Basic dans buffer
 E277 sortir nombre à un octet
 E27D sortir nombre sur deux octets
 E288 sortir numéro de ligne
 E2A3 sortir nombre binaire
 E2AE sortir nombre hexa
 E2C8 sortir nombre à virgule flottante
 E354 adresses des mots instructions Basic
 E388 table des mots instructions Basic
 E64B table des opérateurs Basic
 E676 annuler pointeur de programme
 E69D remplacer adresse de ligne par numéro de ligne
 E6BC convertir ligne d'entrée en code interpréteur
 E6D2 convertir instruction en code interpréteur
 E728 instruction Basic DELETE
 E767 aller chercher adresse de ligne
 E79A chercher ligne Basic
 E7DF instruction Basic RENUM
 E8C1 tester si variable indiquée
 E8EF instruction Basic DATA
 E8F3 instructions Basic ELSE, REM et '
 E9BD instruction Basic RUN
 E9F6 instruction Basic LOAD
 EA3C instruction Basic CHAIN
 EAA6 instruction Basic MERGE
 EC09 instruction Basic SAVE
 EC3D SAVE,P
 EC5C SAVE,B
 EC87 SAVE,A
 EE61 Convertir chiffre ASCII en binaire
 EE79 sortir nombre entier hl
 EE82 convertir nombre entier en ASCII
 EE9D convertir nombre en ASCII
 EE9F formater nombre
 F114 conversion en binaire

F119 conversion en hexa
F158 fonction Basic PEEK
F15F instruction Basic POKE
F16D fonction Basic INP
F177 instruction Basic OUT
F17D instruction Basic WAIT
F194 aller chercher valeurs 16 bits et 8 bits
F1A0 chercher extension d'instruction Basic
F1BA instruction Basic CALL
F1F2 initialiser tabulations
F1F6 instruction Basic ZONE
F1FD instruction Basic PRINT
F25C PRINT,
F277 PRINT SPC
F280 PRINT TAB
F2A0 aller chercher valeur entière entre parenthèses
F2C4 PRINT USING
F3BA tester si caractère de formatage
F47B instruction Basic WRITE
F4C4 configurer la mémoire
F4EF instruction Basic MEMORY
F501 faire de la place pour le programme à charger
F51D calculer longueur de la zone des chaînes
F52C augmenter pointeurs de programme et de variable de bc
F58E initialiser pile Basic
F5A0 libérer place dans pile Basic
F5B0 réserver place dans pile Basic
F5D1 réserver place pour chaîne
F5E6 tester si place dans zone de chaînes
F5F8 réserver place dans zone variables
F618 tester si place dans zone variables
F628 calculer place libre en mémoire
F69D instruction Basic SYMBOL
F6CD SYMBOL AFTER
F7CB lire chaîne
F828 sortir chaîne
F834 fonction Basic LOWER\$
F839 conversion majuscules en minuscules
F842 fonction Basic UPPER\$
F863 addition de chaîne
F897 comparaison de chaîne
F8BA fonction Basic BIN\$

F8C4 fonction Basic HEX\$
F8CE aller chercher arguments pour BIN\$ et HEX\$
F8EA fonction Basic DEC\$
F91E fonction Basic STR\$
F93C fonction Basic LEFT\$
F943 fonction Basic RIGHT\$
F94B fonction Basic MID\$
F993 instruction Basic MID\$
F9E9 aller chercher chaîne et valeur 8 bits
F9FB aller chercher 3ème argument pour MID\$
FA0A fonction Basic LEN
FA10 fonction Basic ASC
FA16 fonction Basic CHR\$
FA24 variable réservée INKEY\$
FA36 fonction Basic STRING\$
FA57 fonction Basic SPACE\$
FA70 aller chercher code ASCII
FA77 fonction Basic VAL
FA92 conversion en entier et test < 256
FAA1 fonction Basic INSTR
FBB3 initialiser pile descripteur
FBDA aller chercher paramètres de chaîne
FC19 réserver place, placer descripteur
FC2D fonction Basic FRE
FC3E Garbage Collection
FCCC opérateur Basic '+'
FCE1 opérateur Basic '-'
FCF5 opérateur Basic '*'
FD09 comparaison arithmétique
FD12 opérateur Basic '/'
FD37 opérateur Basic 'Backslash'
FD49 opérateur Basic MOD
FD58 opérateur Basic AND
FD63 opérateur Basic OR
FD6D opérateur Basic XOR
FD85 fonction Basic ABS
FD89 inverser signe
FDE8 fonction Basic FIX
FDED fonction Basic INT
FE4F convertir opérandes entiers en virgule flottante
FE6A convertir nombre entier en virgule flottante
FE8D fonction Basic CINT

FEC2 fonction Basic UNT
FEEC fonction Basic CREAL
FF02 fonction Basic SGN
FF0A accepter contenu accu comme nombre entier
FF0D accepter nombre entier en hl
FF16 fixer type de variable sur virgule flottante
FF23 aller chercher type de variable
FF27 tester si chaîne
FF2D aller chercher résultat numérique
FF3C tester si chaîne, sinon 'Type mismatch'
FF45 tester si chaîne
FF53 placer résultat sur pile Basic
FF62 copier variable dans (hl)
FF71 tester si lettre
FF8A convertir minuscules en majuscules
FF93 parcourir table
FFAA parcourir table
FFB8 comparer hl <> de
FFBE comparer hl <> bc
FFC4 de:=de-hl
FFCF hl:=hl-de
FFDA bc:=hl-de
FFE7 hl:=hl-bc
FFF2 transfert de bloc ldir
FFF5 transfert de bloc lddr
FFF8 jp (hl)
FFF9 jp (bc)
FFFB jp (de)

4.4 Les tokens Basic

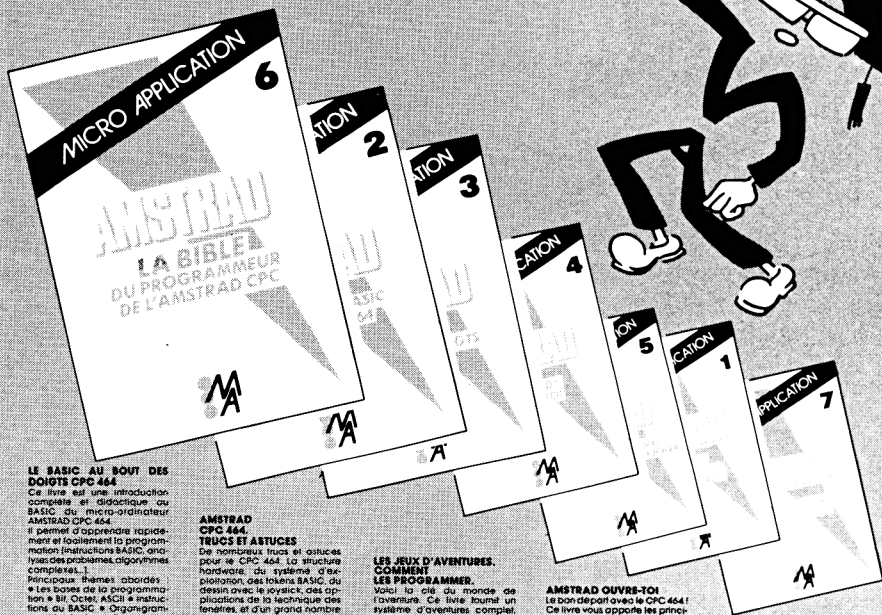
00	Fin de ligne	98	END
01	':', fin de l'instruction	99	ENT
02	variable entière '%'	9A	ENV
03	variable chaîne '\$'	9B	ERASE
04	variable réelle '!'	9C	ERROR
0D	variable sans marque	9D	EVERY
0E	constante 0	9E	FOR
0F	constante 1	9F	GOSUB
10	constante 2	A0	GOTO
11	constante 3	A1	IF
12	constante 4	A2	INK
13	constante 5	A3	INPUT
14	constante 6	A4	KEY
15	constante 7	A5	LET
16	constante 8	A6	LINE
17	constante 9	A7	LIST
19	valeur sur un octet	A8	LOAD
1A	valeur deux octets, décimal	A9	LOCATE
1B	valeur deux octets, binaire	AA	MEMORY
1C	valeur deux octets, hexa	AB	MERGE
1D	adresse de ligne	AC	MIDS
1E	numéro de ligne	AD	MODE
1F	valeur à virgule flottante	AE	MOVE
80	AFTER	AF	MOVER
81	AUTO	B0	NEXT
82	BORDER	B1	NEW
83	CALL	B2	ON
84	CAT	B3	ON BREAK
85	CHAIN	B4	ON ERROR GOTO 0
86	CLEAR	B5	ON SQ
87	CLG	B6	OPENIN
88	CLOSEIN	B7	OPENOUT
89	CLOSEOUT	B8	ORIGIN
8A	CLS	B9	OUT
8B	CONT	BA	PAPER
8C	DATA	BB	PEN
8D	DEF	BC	PLOT
8E	DEFINT	BD	PLOTB
8F	DEFREAL	BE	POKE
90	DEFSTR	BF	PRINT
91	DEG	C0	
92	DELETE	C1	RAD
93	DIM	C2	RANDOMIZE
94	DRAW	C3	READ
95	DRAWR	C4	RELEASE
96	EDIT	C5	REM
97	ELSE	C6	RENUM

C7 RESTORE	FA AND
C8 RESUME	FB MOD
C9 RETURN	FC OR
CA RUN	FD XOR
CB SAVE	FE NOT
CC SOUND	FF Funktion
CD SPEED	CE STOP
CF SYMBOL	
D0 TAG	
D1 TAGOFF	
D2 TRON	
D3 TROFF	
D4 WAIT	
D5 WEND	
D6 WHILE	
D7 WIDTH	
D8 WINDOW	
D9 ZONE	
DA WRITE	
DB DI	
DC EI	
E3 ERL	
E4 FN	
E5 SPC	
E6 STEP	
E7 SWAP	
EA TAB	
EB THEN	
EC TO	
ED USING	
EE >	
EF =	
F0 >=	
F1 <	
F2 <>	
F3 <=	
F4 +	
F5 -	
F6 *	
F7 /	
F8 !	
F9 'Backslash'	

Le token &FF précède une fonction. Il peut être suivi des tokens suivants:

00 ABS	71 BIN\$
01 ASC	72 DEC\$
02 ATN	73 HEX\$
03 CHR\$	74 INSTR
04 CINT	75 LEFT\$
05 COS	76 MAX
06 CREAL	77 MIN
07 EXP	78 POS
08 FIX	79 RIGHT\$
09 FRE	7A ROUND
0A INKEY	7B STRING\$
0B INP	7C TEST
0C INT	7D TESTR
0D JOY	7E 'Improper argument'
0E LEN	7F VPOS
0F LOG	
10 LOG10	
11 LOWER\$	
12 PEEK	
13 REMAIN	
14 SGN	
15 SIN	
16 SPACES\$	
17 SQ	
18 SQR	
19 STR\$	
1A TAN	
1B UNT	
1C UPPERS\$	
1D VAL	
40 EOF	
41 ERR	
42 HIMEM	
43 INKEY\$	
44 PI	
45 RND	
46 TIME	
47 XPOS	
48 YPOS	

AMSTRAD NOUS VOILA!



LE BASIC AU BOUT DES DOIGTS CPC 464

Ce livre est une introduction complète et didactique au BASIC du micro-ordinateur AMSTRAD CPC 464. Il permet d'apprendre rapidement et facilement la programmation (instructions BASIC, environnements, algorithmes complexes).

Principaux thèmes abordés :
 • Les bases de la programmation
 • Les codes ASCII et instructions du BASIC
 • Organigrammes et les fenêtres des Programmes BASIC plus poussés
 • Le programme et menu.
 Contenant de nombreux exemples, ce livre vous assure un apprentissage simple et efficace du BASIC CPC 464.
 Prix : 649 F TTC
 Ref. MA 154

AMSTRAD CPC 464. TRUCS ET ASTUCES

De nombreux trucs et astuces pour le CPC 464. La structure hardware, du système d'exploitation, des logiciels BASIC, du dessin avec le joystick, des applications de la technique des fenêtres, et d'un grand nombre de programmes intéressants tels qu'une gestion de fichier comptable, d'un éditeur de son, d'un générateur de cartes à puces, commandes jusqu'à vos listings complais de vos passionnés.

Prix : 649 F TTC
 Ref. MA 152

LES JEUX D'AVENTURES. COMMENT LES PROGRAMMER.

Voici la clé du monde de l'aventure. Ce livre fournit un système d'aventures complet, avec éditeur, interpréteur, routines attachées et fichiers de jeux. Ainsi qu'un générateur d'aventures pour programmer vous-même facilement vos jeux d'aventures. Avec, bien sûr, des programmes tout prêts à être tapés.

Prix : 629 F TTC
 Ref. MA 123

AMSTRAD OUVRE-TOI

Le bon départ avec le CPC 464 ! Ce livre vous apporte les principales informations sur l'utilisation, les possibilités de connexion du CPC 464 et les rudiments nécessaires pour développer vos propres programmes. C'est le livre idéal pour tous ceux qui veulent pénétrer dans l'univers des micro-ordinateurs avec le CPC 464.

Prix : 99 F TTC
 Ref. MA 120

PROGRAMMES BASIC POUR LE CPC 464

Alimentez votre CPC 464. Ce livre contient de super programmes, notamment un déassembleur, un éditeur graphique, un éditeur de liste. Tous les programmes sont prêts à être tapés et abondamment commentés.

Prix : 189 F TTC
 Ref. MA 119

LA BIBLE DU PROGRAMMEUR DE L'AMSTRAD CPC

LA BIBLE DE L'AMSTRAD CPC est une aide indispensable pour les programmeurs en BASIC et le BASIC absolu pour les programmeurs en assembleur. Cet ouvrage de référence qui révèle vraiment tous les secrets du CPC, est le fruit d'un travail minutieux de plusieurs mois.

Contient :
 - organisation de la mémoire - le processeur particulier du Z 80, du CPC - GAUS ARRAY - le contrôleur vidéo - la ROM vidéo - le CHIP sonore - les interfaces - les systèmes d'exploitation - utilisation des routines avec l'exemple du HARD COPY - le contrôleur de caractères - l'interpréteur BASIC - BASIC et langage machine - le listing de la ROM etc.

Prix : 249 F TTC
 Ref. MA 152

LE LANGAGE MACHINE POUR L'AMSTRAD CPC

Le langage machine pour l'AMSTRAD CPC est fait pour tous ceux qui considèrent que le BASIC n'est plus ni assez puissant ni assez rapide. Des bases de la programmation en langage machine au mode de travail du processeur Z 80 en passant par une description précise de ses instructions, ainsi que l'utilisation des routines systèmes, tout est expliqué complètement et avec de nombreux exemples.

Le livre contient des programmes complets : un assembleur, un débassembleur et un moniteur. Grâce à ce livre le langage machine n'auro plus de secret pour vous.

Prix : 629 F TTC
 Ref. MA 123

MICRO APPLICATION

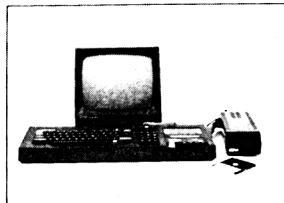
92500 RUEIL-MALMAISON
 147, av. Paul Doumer
 Tél. : (1) 732.92.54
 Telex : MA 205944 F

MICRO

**LES PIRATES,
CES GÉNIES MÉCONNUS!**

**LES PEEKS ET LES POKES
DU COMMODORE 64**

LE COMMODORE PC



DOSSIER : TOUT, TOUT, TOUT SUR LE COMMODORE 128

**EXCLUSIF ! L'ATARI ST
n'est plus qu'à 800 KM.**

CONCOURS AMSTRAD
50 PRIX À GAGNER
Le programme
le plus drôle !

Nous aimerions que l'information circule dans les deux sens !
Cette page vous est réservée.

NOM : ! Matériel utilisé :
Prénom : ! Date d'Achat :
Adresse : ! Extension/Périphérique :
! !
Code Postal : ! Logiciel préféré :
Age : !
Sexe : !

Etes-vous satisfait des logiciels existant ? Oui Non


Si oui, lesquels ?


Si non, quels sont les logiciels que vous aimeriez trouver ?

Que pensez-vous des logiciels MICRO APPLICATION ?

Que pensez-vous des livres MICRO APPLICATION ?

Que pensez-vous de la revue ?

 -----
Votre rubrique personnelle :

 -----
Attention Micro Info n'étant tiré qu'à 10 000 exemplaires.
Réservez dès à présent le numéro spécial Rentrée 85 plus les 3 prochains numéros pour la somme de 60 FF.

Règlement par chèque bancaire ou CCP uniquement.

Bulletin d'abonnement :

NOM :
Prénom :
Adresse :

Code Postal : Ville :

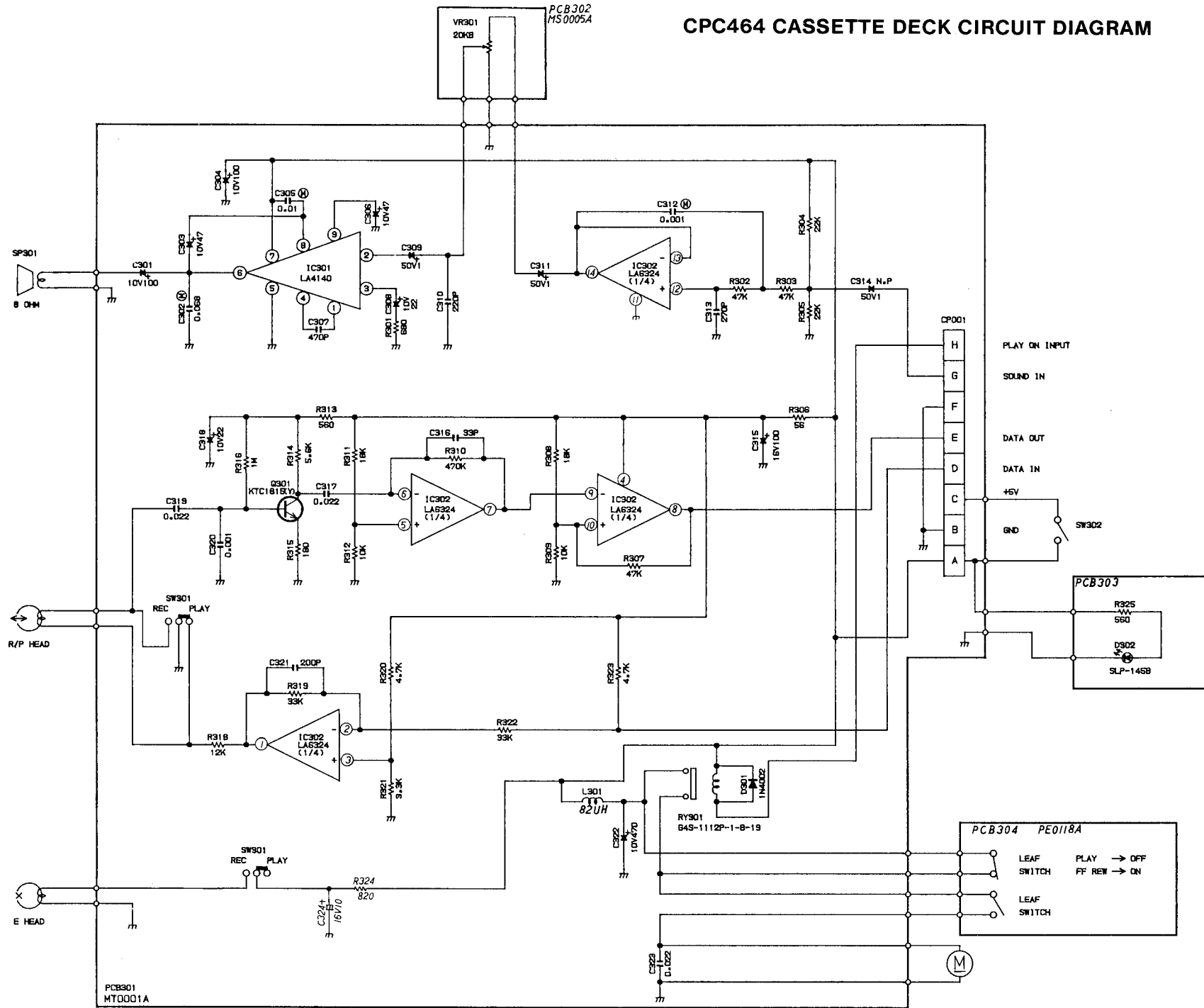
Nos petites Annonces gratuites seront réservées en priorité aux abonnés.

ABONNEMENT

Achévé d'imprimer en juin 1985
sur les presses de l'imprimerie Laballery et C^e
58500 Clamecy
Dépôt légal : juin 1985
N^o d'impression : 505064

PLANS DÉTAILLÉS DE LA MACHINE

CPC464 CASSETTE DECK CIRCUIT DIAGRAM



LA BIBLE DE L'AMSTRAD CPC est une aide indispensable pour les programmeurs en BASIC et le MUST absolu pour les programmeurs en assembleur. Cet ouvrage de référence qui révèle vraiment tous les secrets du CPC, est le fruit d'un travail minutieux de plusieurs mois.

Contenu :

- organisation de la mémoire
- le processeur
- particularité du Z 80, du CPC
- GATE ARRAY
- le contrôleur vidéo
- la ROM vidéo
- le CHIP sonore
- les interfaces
- les systèmes d'exploitation
- utilisation des routines avec l'exemple du HARD COPY
- le générateur de caractères
- l'interpréteur BASIC
- BASIC et langage machine
- le listing de la ROM
- etc.

ISBN : 2-86899-011-8

PRIX : 249 FF

Réf. : ML 122

AMSTRAD

**LA BIBLIE
DU PROGRAMMEUR
DE L'AMSTRAD CPC**



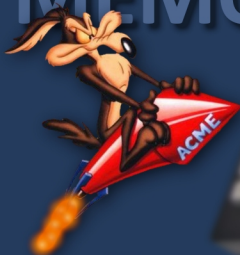


Document numérisé
avec amour par :

AMSTRAD

CPC 

MÉMOIRE ÉCRITE



<http://amstradcpc.fredisland.net/>