

D. ROY – J.J. WEYER

MICRO APPLICATION

14

AMSTRAD

**LES ROUTINES DE L'AMSTRAD
CPC 464, 664 ET 6128**



UN LIVRE DATA BECKER

D. ROY — J.J. WEYER

MICRO APPLICATION

14

AMSTRAD

LES ROUTINES DE L'AMSTRAD
CPC 464, 664 ET 6128



— UN LIVRE DATA BECKER —

Distribué par : MICRO APPLICATION
13, Rue Sainte Cécile
75009 PARIS

et

EDITION 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, article 40, 1er alinéa).

Cette représentation ou reproduction illicite, par quelque procédé que ce soit, constituerait une contrefaçon sanctionnée par les articles 425 et suivants de 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 : 2-86899-026-6

(c) 1985 MICRO APPLICATION
13 Rue Sainte Cécile
75009 PARIS

édité par Frédérique BEAUDONNET

INTRODUCTION DE L'OUVRAGE:

I - OBJET:

Cet ouvrage traite de l'utilisation des routines mises à la portée du programmeur de l'amstrad; celles-ci d'un emploi relativement simple, permettront une programmation assembleur plus aisée.

Il nous a semblé important de développer cet aspect de la programmation, que tout utilisateur abordera un jour ou l'autre, qu'il soit débutant ou chevronné.

De nos jours, de plus en plus d'amateurs cherchent à aller plus loin dans la maîtrise des langages sur micro-ordinateur, et de ce fait souhaitent passer du basic à l'assembleur; ce langage plus proche de la machine leur offre une amélioration considérable de la rapidité des logiciels, et en particulier au niveau des jeux. Le lecteur désireux de se perfectionner, trouvera dans cet ouvrage une panoplie importante de routines sur les diverses fonctions qui feront de l'amstrad un outil informatique très apprécié, tant au niveau professionnel, qu'au niveau familial.

II - CONCEPTION DE L'OUVRAGE:

Cet ouvrage a été conçu méthodiquement, à l'aide de tests personnalisés, apportant ainsi à l'utilisateur une foule de renseignements pour chacune des routines traitées. De nombreux exemples étoffent la plupart des chapitres, essayant d'être le plus exhaustifs possible.

III - PRESENTATION DES CHAPITRES:

Le lecteur peut aborder cet ouvrage par le chapitre qui lui convient, le plan du livre respectant l'indépendance relative des différents chapitres, exception faite toutefois pour les 3 chapitres (5 à 7) concernant l'affichage 'texte' ou 'graphique' et gestion d'écran.

Le chapitre 1 aborde succinctement la description matérielle du micro-ordinateur amstrad; ce chapitre a pour unique objectif de familiariser le lecteur avec les différents éléments du système.

Le chapitre 2 quant à lui présente l'aspect organisation de la mémoire de l'amstrad, aussi bien pour l'espace utilisateur que pour l'espace système, et que l'on appelle respectivement RAM et ROM.

Nous arrivons, avec le chapitre 3, à des notions d'informatique plus précises, pour entrer dans la structure d'un programme basic en mémoire, en passant par la construction des variables utilisées.

A partir du chapitre 4 commence véritablement l'étude des routines. Dans ce premier d'une série de chapitres présentés de façon similaire, il est fait état des diverses routines du clavier. A la fin de ce chapitre le lecteur doit être en mesure de procéder à la redéfinition des touches du clavier, de créer des fonctions qui seront rapportées au pavé numérique, et de changer à volonté la vitesse de répétition des touches du clavier.

Avec les chapitres 5 à 7, le lecteur aborde une des plus importante fonction de l'amstrad, soit l'affichage à l'écran; que ce soit du texte ou du graphisme, il devra gérer judicieusement la couleur, le tracé, la création de fenêtres (espaces plus ou moins réduits d'affichage), et également la redéfinition des caractères.

Le chapitre 5 présente tout d'abord une routine importante du système qui permet des affichages sophistiqués avec peu d'instructions; les routines qui sont étudiées ensuite, traitent de l'affichage des caractères existants ou redéfinis.

Le chapitre 6 reprend à peu de chose près le même cheminement que le chapitre 5, à la différence qu'ici il est question de graphisme, et que de ce fait nous parlerons de points (pixels) au lieu de caractères (matrices de points).

Avec le chapitre 7 nous entrons véritablement dans la gestion de l'écran; ici l'artiste est roi car il peut directement exécuter ses oeuvres graphiques en 'mémoire écran'; pour cela il lui est offert la possibilité d'intervenir au niveau de la trame même de l'écran, et au niveau le plus fin de la représentation des points; en ce qui concerne le codage des couleurs nous lui proposons des tableaux pour chacun des modes, permettant, avec une relative facilité, la création de dessin superbement colorés.

Le chapitre 8 est articulé autour de l'unité de cassette; les passionnés en matière de gestion de fichiers, trouveront tout sur la création des fichiers (normalisés ou particuliers); à la fin de ce chapitre il leur sera également possible de se constituer des listes d'informations complètes pour chacun de leurs fichiers.

Les fanas du système trouveront quant à eux, dans le chapitre 9, le moyen de se connecter sur le système d'exploitation (ROM basse), ou sur l'interpréteur basic (ROM haute); des routines de transfert de blocs d'octets leur sont également proposées. Un certain nombre d'annexes est proposé à l'utilisateur; celles-ci ont pour dessein d'être complémentaires ou supplémentaires des divers chapitres traités tout au long de l'ouvrage.

L'annexe 1 propose des tables permettant de retrouver n'importe quelle instruction assembleur, à partir de son code instruction.

En annexes 2 et 3 vous trouverez une liste complète du système d'exploitation et de l'interpréteur basic; cette liste a été conçue de façon à tenir le moins de place possible dans l'ouvrage, tout en offrant une relative aisance quant à sa consultation.

L'annexe 4 est un recueil de 5 programmes basic et assembleur; leur utilisation doit s'avérer bénéfique pour la compréhension de certains des chapitres évoqués plus haut, mis à part le programme de sauvegarde de logiciels qui est plutôt proposé comme un utilitaire.

Enfin, pour les mordus du 'vocabulaire', l'ouvrage se termine par un mini lexique recensant les principaux termes techniques informatiques mentionnés dans ce livre.

IV - MISE EN PRATIQUE:

Pour qu'un sujet soit bien assimilé il est nécessaire voir même indispensable de faire des exercices, et à fortiori de bien assimiler les exemples qui sont exposés; pour cela vous trouverez en annexe 4 un programme basic permettant de charger les exemples qui mettent en évidence les éléments de chacun des chapitres où ils sont présents; ce logiciel que nous dénommons 'dump aux fonctions multiples' permet d'entrer des petits programmes assembleurs (codes hexadécimaux), de les sauvegarder sur cassette ou sur disquette, puis de les relire ultérieurement.

Ce logiciel offre également la possibilité d'entrer du texte en mémoire (entrée ASCII), puis de procéder à l'affichage de celui-ci, ou d'une liste de codes hexa; il suffit pour cela de sélectionner le menu 3 (DUMP), en ce qui concerne la mémoire RAM.

PRECISIONS SUR LA STRUCTURE DES EXEMPLES EN ASSEMBLEUR:

Une première partie donne la liste (listing) du programme assembleur; une seconde partie éventuelle présente les données utilisées par le programme (textes, codes de commande). Le programme assembleur est présenté sur 5 colonnes, dont certaines ne sont pas toujours utilisées.

La première colonne indique les adresses où sont implantées chacune des instructions en mémoire; dans tous nos exemples nous avons volontairement défini l'adresse de début en &7000.

La seconde colonne donne les codes hexa des instructions du programme; ce sont ces codes qu'il faut entrer au niveau du programme 'dump aux fonctions multiples'.

Les troisième et quatrième colonnes forment l'instruction sous la forme assembleur; ce sont ces instructions que vous utiliserez si vous disposez d'un assembleur-désassembleur;

la troisième colonne indique le label,

la quatrième colonne l'instruction elle-même.

La cinquième colonne, commençant par un point-virgule, n'est autre que des commentaires, destinés à expliciter le programme.

Les programmes commencent par des instructions, dont les première et seconde colonnes n'existent pas; ces instructions ne sont utilisées qu'avec un assembleur-désassembleur.

ORG indique l'origine du programme et données en mémoire.

LOAD indique l'adresse où est effectivement chargé le programme.

EQU donne des adresses appelées par CALL ou LD (dans nos ex.) Les données sont présentées sous forme de DUMP, c'est à dire sous la forme d'une liste de codes hexa, dont les codes ASCII sont traduits en caractères dans une colonne séparée. Les adresses de début des différents textes sont indiquées au niveau du programme assembleur; pour que celui-ci puisse fonctionner, il faut également rentrer les données en mémoire; ceci se fera également à l'aide du logiciel 'dump aux fonctions multiples'. Toutes les données correspondant à des textes en codes ASCII, pourront être entrées par le menu (entrée ASCII); toutes les autres données devront être entrées avec le menu (entrée codes machine).

Nous espérons qu'à l'aide de ces exemples cet ouvrage vous donnera l'envie d'aller plus loin dans la découverte de l'amstrad.

=== LA MAITRISE DE L'AMSTRAD ===

=====

== SOMMAIRE ==

=====

INTRODUCTION

=====

CHAPITRE 1 DESCRIPTION HARDWARE DE L'AMSTRAD

=====

I	————— Le micro-processeur	1
II	————— L'unité logique	1
III	—— Le controleur de tube cathodique	2
IV	——— L'interface d'entrée-sortie	2

CHAPITRE2...ORGANISATIONDELA MEMOIREDEL'AMSTRAD

=====

I	————— La mémoire ROM	4
---	----------------------	---

II	————— La mémoire RAM	5
III	- Carte memoire ROM et RAM de l'amstrad	7

CHAPITRE 3 STRUCTURE D'UN PROGRAMME BASIQUE EN MEMOIRE

I	————— Le programme	8
II	————— Les variables	18

CHAPITRE 4 GESTION DU CLAVIER

I	————— Généralités	28
II	————— Routines utilisées	29
III	————— Exemples d'utilisation	51

CHAPITRE 5 L'ECRAN TEXTE

I	————— Généralités	56
II	————— La routine ROM	57
III	— Les routines utilisées	71

CHAPITRE 6 L'ECRAN GRAPHIQUE

=====

I ————— Généralités	114
II — Les routines utilisées	115
III — Exemple d'utilisation	138

CHAPITRE 7 . GESTION DE LA MEMOIRE ECRAN

=====

I ————— Généralités	142
II — Les routines utilisées	159
III — Exemple d'utilisation	194

CHAPITRE 8 L'UNITE DE CASSETTE

=====

I ————— Généralités	202
II Organisation d'un fichier	202
III — Les routines utilisées	205

IV — Exemple d'utilisation	224
----------------------------	-----

CHAPITRE 9 ROUTINES "SYSTEME"

=====

I ————— Généralités	227
II ————— Routines utilisées	228

ANNEXE 1 . TABLES INVERSES DES INSTR Z80

=====

I ————— Présentation	236
II ————— Tables inverses	237

ANNEXE 2 PROGRAMMES DIVERS

=====

I ————— Présentation	248
II ————— Décimal → binaire	251
III ————— Analyse Registres	252
IV ——— Sauvegarde de logiciels	253
V — Dump aux fonctions multiples	257

CHAPITRE 1 -

DESCRIPTION HARDWARE DE L'AMSTRAD

Bien que cette description ne soit pas nécessaire pour se lancer dans la programmation, elle permet cependant de mieux appréhender le déroulement de certaines instructions, et notamment au niveau de la programmation en assembleur.

I - LE MICRO-PROCESSEUR:

L'amstrad est doté d'un micro-processeur 8 bits, ce qui implique qu'il peut adresser 64K maximum de mémoire à un moment donné. Le micro-processeur qui équipe l'amstrad est le célèbre Z80A, dont l'horloge interne a un quartz qui oscille à quatre méga-hertz. Le Z80A est un micro-processeur qui permet une puissance de calcul assez impressionnante, et qui concurrence sans conteste d'autres micro-processeurs équipant des systèmes bien plus importants.

II - L'UNITE LOGIQUE:

L'unité logique de l'amstrad est également une partie importante puisque celle-ci s'occupe de la gestion de contrôle du système. Elle permet notamment, en liaison avec ses satellites (contrôleur de tube cathodique, unité centrale, et mémoire RAM), de gérer le mode écran ainsi que les couleurs, et de générer les signaux vidéo utilisés par le moniteur de l'amstrad.

III - LE CONTROLEUR DE TUBE CATHODIQUE:

Le controleur de tube cathodique de l'amstrad (CRTC 6845) a pour fonction, en liaison avec l'unité logique, de gérer les signaux vidéo qui seront ensuite envoyés au moniteur.

IV - L'INTERFACE D'ENTREE - SORTIE :

L'interface d'entrée-sortie revêt une importance particulière, puisqu'elle permet au micro-ordinateur de communiquer avec les périphériques internes et externes. Pour ce qui concerne l'amstrad, les périphériques internes sont limités au seul générateur programmable de son qui n'est autre que le célèbre AY-3-8912. Nous pouvons par contre dénombrer une grande quantité de périphériques externes utilisables, dont le lecteur de cassette, qui lui est intégré au boîtier de l'amstrad.

En ce qui concerne les périphériques externes, l'amstrad permet la connection d'une imprimante, via le port parallèle centronics; il est également possible de connecter une unité de disquette, et de brancher des manettes de jeu. La fonction d'interfaçage, de certains des périphériques internes ou externes, est assurée par la puce PPI 8255 (parallel peripheral interface) qui est dotée de trois ports d'entrée-sortie.

(1) PORT A:

Le port A permet la communication avec le générateur programmable de son AY-3-8912, dont les données sont envoyées sur huit bits.

(2) PORT B:

Le port B a une double fonction : la gestion des signaux devant parvenir au port parallèle centronics et la lecture des données venant de l'unité de cassette.

(3) PORT C:

Le port C a de multiples fonctions. Il permet d'une part de contrôler le moteur de l'unité de cassette et de transférer des données sonores au PSG et permet, d'autre part, la fonction d'écriture des données sur les cassettes et la lecture des touches du clavier par sélection des rangées suivant un matriçage défini par le système.

CHAPITRE 2 - ORGANISATION DE LA MEMOIRE DE L'AMSTRAD

I - LA MEMOIRE ROM

La ROM (read only memory) est un type de mémoire qui ne peut être que lu ; la ROM est, dans tous les micro-ordinateurs, utilisée pour contenir le cœur du système. Dans le cas de l'amstrad la ROM est en overlay, c'est à dire que celle-ci ne peut être adressée en même temps que la RAM ; en effet la ROM et la RAM sont situées aux mêmes adresses d'accès.

Il est par conséquent nécessaire d'utiliser une méthode d'accès qui puisse passer de l'une à l'autre sans aucune difficulté. Pour cela il est indispensable que les routines utilisées soient à la fois dans la ROM et dans la RAM, et de surcroît aux mêmes adresses mémoire; ces routines se situent aux adresses 0 à 3CH de la RAM et de la ROM ; celles-ci sont appelées "RESTARTS" et sont au nombre de 8 (RST 0 à 7) ; elles sont abondamment utilisées par le système car elles permettent le "va et vient" entre ROM et RAM.

ROM BASSE et ROM HAUTE:

La ROM de l'amstrad est scindée en deux parties de 16k chacune.

La première partie est située aux adresses 0000H à 3FFFH ; celle-ci contient le système d'exploitation de l'amstrad; les adresses 000H à 003CH contenant les routines d'accès ROM <→ RAM.

La deuxième partie, quant à elle, contient l'interpréteur basic, et se situe aux adresses C000H à FFFFH. La première partie est appelée ROM "basse" (lower ROM), et la deuxième partie est appelée ROM "haute" (upper ROM).

II - LA MEMOIRE RAM

La RAM (random access memory) est un type de mémoire qui peut être, ou lu, ou écrit; la RAM est dans tous les micro-ordinateurs, une mémoire destinée à l'utilisateur; mais celle-ci peut également être utilisée par le système. Pour l'amstrad, souvenez-vous de ce qui a été dit pour la ROM, la zone mémoire située aux adresses 000H à 003CH est aussi utilisée pour les routines d'accès RAM $\leftrightarrow$ ROM.

Le système se sert également d'autres parties de la RAM, dont la plus importante en place est la zone de mémoire de l'image écran située aux adresses C000H à FFFFH.

La zone mémoire B900H à C000H est aussi utilisée par le système, pour y loger des routines indispensables pour le programmeur en langage assembleur; celles-ci seront abondamment traitées tout au long des chapitres de cet ouvrage.

Enfin, c'est là ce qui intéresse plus spécialement le programmeur de l'amstrad, la zone utilisateur (basic et assembleur) se situe aux adresses 0170H à AB7FH. Les zones permettant la manipulation de certaines données, sont situées aux adresses 0040H à 016FH et AB80H à B8FFH.

ROUTINES D'ACCES RAM <=> ROM (RESTARTS)

ROUTINE	ADRESSE	INSTR	FONCTION	UTILISATION
RST	HEXA	APPEL		

0	00	C7	initialisation	CALL 0 / RST 0
1	08	CF	accès ROM B / RAM	RST 08H
2	10	D7	accès ROM extension	RST 10H
3	18	DF	accès RAM / ROM	RST 18H
4	20	E7	accès RAM	RST 20H
5	28	EF	accès ROM B	RST 28H
6	30	F7	routine utilisateur	RST 30H
7	38	FF	gère interruptions	RST 38H

D'autres ouvrages ayant déjà traité de l'utilisation des RESTARTS, nous nous limiterons à cette présentation succincte.

III - CARTE MEMOIRE ROM ET RAM DE L'AMSTRAD

FFFFH	image mémoire de l'écran	FFFFH	interpréteur basic
C000H	routines pour l'utilisateur	C000H	
B900H	zone de données (système)		
AB80H	zone utilisateur pour programmes basic et assembleur		
0170H	zone de données (système)	3FFFH	système d'exploitation
0040H	restarts 0 a 7 (système)		restarts 0 a 7 (système)
0000H		0000H	

CHAPITRE 3 - STRUCTURE D'UN PROGRAMME BASIC EN MEMOIRE

I - LE PROGRAMME

Tout programme basic commence à l'adresse 0170H de la mémoire RAM.
La structure d'une instruction basic en mémoire est la suivante:

(1) octets 1 et 2 d'une instruction:

Ces deux octets indiquent la longueur de l'instruction, afin que le système puisse savoir où débute l'instruction suivante.

(2) octets 3 et 4 d'une instruction:

Ces deux octets indiquent le numéro de ligne de l'instruction basic; ce numéro étant codé sur 2 octets, nous ne pouvons avoir que des lignes numérotées de 0 à 65535.

(3) octets 5 à la fin de l'instruction:

Ces octets représentent l'instruction elle-même, que nous allons détailler sur un exemple complet.

(4) dernier octet de l'instruction:

Celui-ci est toujours égal à zéro; son but est d'indiquer au système que l'instruction se termine ici.

Revenons aux octets suivants (3):

Soit l'instruction basic → 10 FOR I=1 TO 20

Pas de surprise pour les quatres premiers octets, nous aurons:

< 0F 00 > < 0A 00 >

En ce qui concerne les octets 5 à la fin de l'instruction nous aurons:

<	9E	0D	05	00	C9	EF	0F	EC	19	14	>
a	b	c	d	e	f	g	h	i	j		

L'octet (a) est le code de l'instruction basic FOR.

Les octets (b) à (e) représentent la variable I; nous allons voir plus loin ce qu'ils signifient.

L'octet (f) est le code de l'instruction basic '='.

L'octet (g) représente la valeur numérique 1.

L'octet (h) est le code de l'instruction basic TO.

Les octets (i) et (j) représentent la valeur numérique de 20.

La représentation des valeurs numériques des variables sera expliquée en détail dans ce manuel.

FIN D'UN PROGRAMME BASIC:

Un programme basic se termine par cinq octets à zéro. Ceci est tout à fait logique; en effet, la dernière instruction se termine par un zéro; puisque le système cherche à se brancher sur une prochaine instruction, la seule façon de l'arrêter est de lui indiquer une longueur d'instruction nulle (2 octets à 00), et de lui indiquer un numéro de ligne basic nul (2 octets à 00).

LES CODES INSTRUCTION BASIC:

Pour les découvrir, il suffit de taper les instructions basic, en indiquant seulement une instruction par ligne, puis de faire un dump de cette suite de lignes d'instructions; ceci est long et fastidieux; aussi je vous propose une liste alphabétique, et une table inverse des codes instruction basic. Vous trouverez en annexe, un programme basic de dump qui vous permettra de mettre en pratique les éléments de ce chapitre, mais qui pourra être aussi utilisé pour d'autres applications.

Liste alphabétique des instructions basic avec leur code hexa.

Cette liste peut également servir de table inverse pour retrouver une instruction basic suivant son code hexadécimal ou décimal.

1) Instructions dont le code est sur un octet:

-----			-----			-----		
INSTRUCTION	DEC	HEX	INSTRUCTION	DEC	HEX	INSTRUCTION	DEC	HEX

After	128	80	! Auto	129	81	! Border	130	82
Call	131	83	! Cat	132	84	! Chain	133	85
Clear	134	86	! Clg	135	87	! Close in	136	88
Closeout	137	89	! Cls	138	8A	! Cont	139	8B
Data	140	8C	! Def	141	8D	! Defint	142	8E
Defreal	143	8F	! Defstr	144	90	! Deg	145	91
Delete	146	92	! Dim	147	93	! Draw	148	94
Drawn	149	95	! Edit	150	96	! Else	151	97
End	152	98	! Ent	153	99	! Env	154	9A
Erase	155	9B	! Error	156	9C	! Every	157	9D
For	158	9E	! Gosub	159	9F	! Goto	160	A0
If	161	A1	! Ink	162	A2	! Input	163	A3
B Key	164	A4	! Let	165	A5	! Line	166	A6
List	167	A7	! Load	168	A8	! Locate	169	A9
Memory	170	AA	! Merge	171	AB	! Mid\$	172	AC
Mode	173	AD	! Move	174	AE	! Mover	175	AF
Next	176	B0	! New	177	B1	! On	178	B2
On break	179	B3	! On err goto	180	B4	! On sq	181	B5
Openin	182	B6	! Openout	183	B7	! Origin	184	B8
Out	185	B9	! Paper	186	BA	! Pen	187	BB

Plot	188 BC	!	Plotr	189 BD	!	Poke	190 BE
Print	191 BF	!	'	192 C0	!	Rad	193 C1
Randomize	194 C2	!	Read	195 C3	!	Release	196 C4
Rem	197 C5	!	Renam	198 C6	!	Restore	199 C7
Resume	200 C8	!	Return	201 C9	!	Run	202 CA
Save	203 CB	!	Sound	204 CC	!	Speed	205 CD
Stop	206 CE	!	Symbol	207 CF	!	Tag	208 D0
Tagoff	209 D1	!	Troff	210 D2	!	Tron	211 D3
Wait	212 D4	!	Wend	213 D5	!	While	214 D6
Width	215 D7	!	Window	216 D8	!	Write	217 D9
Zone	218 DA	!	Di	219 DB	!	Ei	220 DC
Erl	227 E3	!	Fn	228 E4	!	Spc	229 E5
B Step	230 E6	!	Swap	231 E7	!	Tab	234 EA
Then	235 EB	!	To	236 EC	!	Using	237 ED
>	238 EE	!	=	239 EF	!	>=	240 F0
<	241 F1	!	<>	242 F2	!	<=	243 F3
+	244 F4	!	-	245 F5	!	*	246 F6
/	247 F7	!	^	248 F8	!	\	249 F9
And	250 FA	!	Mod	251 FB	!	Or	252 FC
Xor	253 FD	!	Not	254 FE	!		

2) Instructions dont le code est sur 2 octets:

La plupart des micro-ordinateurs proposent une panoplie d'instructions dont les codes instruction sont en principe compris entre 80H et FFH.

L'amstrad propose un nombre assez important d'instructions basic et en se reportant au tableau des instructions dont le code est sur un octet, on constatera que le code FFH n'est pas utilisé; celui-ci a en effet été réservé pour agrandir le parc des instructions basic de l'amstrad. Ces instructions sont composées essentiellement d'ordres ayant pour but le traitement des variables et valeurs numériques, ainsi que les chaînes de caractères.

Ces instructions auront donc comme premier octet du code → FFH
Le deuxième octet étant défini suivant une liste et table inverse.

TABLE INVERSE DES INSTRUCTIONS BASIC SUR 2 OCTETS

INSTRUCTION DEC HEX ! INSTRUCTION DEC HEX ! INSTRUCTION DEC HEX

Abs	0	00	!	Asc	1	01	!	Atn	2	02
Chr\$	3	03	!	Cint	4	04	!	Cos	5	05
Creal	6	06	!	Exp	7	07	!	Fix	8	08
Fne	9	09	!	Inkey	10	0A	!	Inp	11	0B
Int	12	0C	!	Joy	13	0D	!	Len	14	0E
Log	15	0F	!	Log10	16	10	!	Lower\$	17	11
Peek	18	12	!	Remain	19	13	!	Sgn	20	14
Sin	21	15	!	Space\$	22	16	!	Sq	23	17
Sqr	24	18	!	Str\$	25	19	!	Tan	26	1A
Unt	27	1B	!	Upper\$	28	1C	!	Val	29	1D
Eof	64	40	!	Err	65	41	!	Himem	66	42
Inkey\$	67	43	!	Pi	68	44	!	Rnd	69	45
Time	70	46	!	Xpos	71	47	!	Ypos	72	48
Bin\$	113	71	!	Dec\$	114	72	!	Hex\$	115	73
Instr	116	74	!	Left\$	117	75	!	Max	118	76

APPLICATION PRATIQUE:

Nous pouvons d'ores et déjà imaginer un bon nombre d'applications pratiques, basées sur "l'auto-modification" d'un programme basic. Nous avons tous été confrontés au problème qui consiste à modifier une instruction à l'aide d'une programmation basic; mais combien décevants ont été les résultats ! Je pense notamment à l'entrée de fonction, qui actuellement oblige le programme basic à revenir en ready, et à retaper l'instruction où se trouve la fonction pour y mettre une nouvelle fonction...

Après ce qui vient d'être expliqué sur le basic en mémoire de l'amstrad nous sommes cette fois en mesure de remédier à ce problème en utilisant l'instruction POKE; mais prudence car la moindre faute pourra entraîner des destructions partielles de votre programme basic.

Afin de bien fixer les idées, je vous propose un court programme basic dont le but est de changer tour à tour une fonction SIN en LOG, et réciproquement, si le résultat $y=f(x)$ est nul.

```
10 CLS:DEG
20 INPUT "X=";X
30 Y=SIN(X)
40 IF PEEK(401)=15 THEN A$="LOG"
   ELSE A$="SIN"
50 PRINT A$ "("X")="Y
60 IF X=0 THEN POKE 401,15
70 IF X=1 THEN POKE 401,21
80 GOTO 20
```

A chaque fois que y est égal à zéro la fonction est changée, soit en SIN si nous étions en LOG, soit en LOG si nous étions en SIN. Pour cela il faut connaître exactement où se situe la valeur hexa de la fonction SIN ou LOG; pour ce faire, une fois le programme tapé et dûment vérifié, nous pouvons taper l'ordre "direct" —>

FOR I=&170 TO &200:IF HEX\$(PEEK(I))=15

THEN PRINT I:END ELSE NEXT

Ceci aura pour effet d'afficher le nombre 401, qui correspond à l'emplacement mémoire de l'instruction basic SIN; cependant il faut être très vigilant quant à l'emploi de cet ordre direct. En effet, il faut s'assurer que dans les instructions avant celle où se trouve la fonction SIN, il n'y a pas d'autre valeur 15. Si le cas se présentait il faudrait alors commencer par situer l'emplacement de l'instruction comportant la fonction SIN. Nous aurions alors, dans notre exemple, bien qu'ici cela ne se justifie pas —>

FOR I=&170 TO &200:A=PEEK(I)+PEEK(I+1)*256:

IF A=30 THEN PRINT I: END ELSE NEXT

Cet ordre afficherait alors l'adresse du numéro de l'instruction basic 30; soit n la valeur obtenue, il suffit alors de taper l'ordre suivant pour obtenir l'emplacement de la fonction SIN —>

```
FOR I=n TO &200:IF HEX$(PEEK(I))=15  
THEN PRINT I:END ELSE NEXT
```

Nous obtiendrons alors à nouveau la valeur 401; maintenant le reste du programme s'explique de lui-même...

II - LES VARIABLES

Revenons à notre exemple de départ: 10 FOR I=1 TO 20
Les octets (b) à (e) représentent la variable I comme suit:
Rappel des octets (b) à (e): < 0D 05 00 C9 >

(1) octet (b):

Ce premier octet a pour but d'identifier la variable; c'est le type de variable (entière, réelle, chaîne).

TYPE DE VARIABLE	!	VALEUR
Réelle	!	0DH
Chaîne	!	03H
Entière	!	02H

(2) octets (c) et (d):

Ces deux octets indiquent l'emplacement de la variable dans la zone variable. Il faut savoir en effet que les variables sont situées juste derrière le programme basic, dans une zone que nous appellerons "zone variable".

Ces deux octets permettent donc au système de retrouver les variables très facilement, en sachant, rappelez vous, qu'un programme basic se termine par cinq zéros. Vous comprendrez maintenant pourquoi le fait de modifier un programme basic interdit de continuer son exécution, à partir de variables précédemment manipulées.

(3) octet (e):

Dans cet exemple nous avons affaire à un nom de variable sur un caractère; aussi afin de bien comprendre le mécanisme de codage d'un nom de variable en mémoire nous allons choisir un nom de variable sur trois octets, soit: ABI Le codage mémoire de ce nom de variable, en mémoire, nous donnerait les octets suivants:

< 41 42 C9 >

e1 e2 e3

On remarquera très facilement que les 2 premiers octets, (e1) et (e2) représentent les codes ASCII de A et B. En ce qui concerne le dernier caractère, soit I, nous avons la valeur C9, qui n'est autre que le code ASCII de I, soit 49, et auquel il a été ajouté 80H.

C'est cette méthode qui permet au système de déterminer la fin du nom de la variable; mais pourquoi le système ajoute t-il 80H au dernier caractère d'un nom de variable ? Tout simplement parce que cette valeur ne change pas la valeur ASCII du caractère (en théorie), car le système ne se sert jamais du dernier bit pour la détermination des caractères. La raison en est très simple; les codes ASCII des caractères prennent les valeurs 20H à 7FH, ce qui signifie que le premier quartet de l'octet d'un code ASCII ne dépasse jamais 7, et que le système n'utilise que les 3 bits de droites; en ajoutant 80H à l'octet, nous ne changeons pas la valeur du code ASCII.

Prenons un exemple: soit le code ASCII 49H (caractère I).

Le premier quartet est 4, soit 0100 en binaire, et le deuxième est 9; en ajoutant 80H à 49H, il est ajouté 0 à 9 et, 8 à 4. Quand la valeur 8 est ajoutée au premier quartet de l'octet, seul le premier bit de gauche est modifié, passant de 0 à 1; les trois bits de droite restant alors inchangés. A l'interprétation d'une instruction basic, et notamment au moment de l'identification d'un nom de variable, le système procède en deux phases:

a) Phase de détection de la fin du nom de la variable:

C'est ici que le système va examiner la valeur du premier bit à gauche du premier quartet; si celui-ci est à zéro il décrète que le nom est incomplet, et il continue sa recherche; si celui-ci est à un, le système sait qu'il est en présence du dernier caractère du nom; c'est à ce moment que le système passe à la seconde phase.

b) Phase de repérage du caractère:

Cette phase est également exécutée après chaque détection des premiers caractères du nom. Dans cette phase le système repère le code ASCII du caractère, uniquement en décodant les trois bits de droite du premier quartet, et les quatre bits du deuxième quartet.

(4) octet (g) et (i) (j):

l'octet (g) représente la valeur numérique 1, alors que les octets (i) et (j) représentent la valeur numérique 20. On s'aperçoit déjà que pour certaines valeurs différentes le codage est différent, et pour un nombre d'octets différent.

En fait ce codage est plus ou moins complexe, selon que nous sommes en présence de nombres positifs ou négatifs, de nombres entiers ou flottants (décimaux).

a) Nombres entiers positifs:

Premier groupe de nombres → les chiffres 0 à 9:

Ils sont codés de 0E à 17; dans l'exemple que nous avons pris au départ, nous avons 0E pour l'octet (g), ce qui correspond bien à la valeur numérique 1.

Deuxième groupe de nombres → les nombres 10 à 255:

Ces nombres sont codés sur 2 octets, le premier octet a pour valeur hexa 19. Le deuxième octet n'est autre que la valeur hexa

du nombre; nous pouvons donc avoir les valeurs hexa 0A à FF.

Dans notre exemple de départ nous avons < 19 14 > pour la valeur 20, et nous constatons que cette valeur hexa est correcte.

Troisième groupe de nombres → les nombres 256 à 32767:

Ces nombres sont codés sur 3 octets, le premier octet ayant pour valeur hexa 1A. Les deux octets suivants représentant la valeur hexa du nombre, mais ceux-ci étant inversés, bien évidemment.

EXEMPLE: le nombre 32767 sera codé < 1A FF 7F >

Quatrième groupe de nombres → les nombres 32768 à 65535:

Ces nombres sont codés sur 6 octets; les trois premiers octets ont pour valeur hexa < 1F 00 00 >; le sixième octet ayant quant à lui la valeur hexa 90. Les octets 4 et 5 représentent la valeur hexa du nombre moins la valeur hexa 8000; pour trouver la valeur décimale du nombre, il faut donc ajouter, avant traduction, la valeur hexa 80 au cinquième octet.

EXEMPLE: le nombre 65000 sera codé < 1F 00 00 E8 7D 90 >

b) nombres entiers négatifs:

La codification des nombres est à peu près identique à celle des nombres positifs; la seule différence réside dans le fait qu'un octet de valeur hexa F5 est mis devant les autres octets. En rajoutant cet octet supplémentaire pour les nombres négatifs, le système de l'amstrad s'octroie la possibilité de traiter des nombres négatifs entiers de -1 à -65535.

c) nombres flottants positifs:

Pour représenter les nombres flottants en mémoire, le système de l'amstrad a besoin de 6 octets utilisés comme suit:

premier octet → valeur hexa 1F

octets 2 à 5 → mantisse du nombre

dernier octet → exposant du nombre

Pour la compréhension du codage de ceux-ci, il est indispensable de connaître la représentation binaire d'un nombre négatif. Aussi, en guise de rappel pour ceux qui le connaissent déjà, et surtout pour ceux qui le découvriront, je vous propose de faire un petit retour en arrière dans le système de numération à base 2.

Exemple: soit le nombre décimal flottant 10.5, sa représentation binaire est → .1010.1000

Les 4 bits avant le point représentent la partie entière du nombre. Les bits qui sont situés derrière le point représentent la partie décimale du nombre. Voyons comment à partir de ce nombre binaire, nous pouvons avoir son équivalent en base décimale.

```

nous avons ----->      1 0 1 0 . 1 0 0 0
                           ! ! ! !   ! ! ! !
                           ! ! ! !   ! ! ! !

1 x 2 puissance 3 ----- ! ! !   ! ! !   --- 0 x 2 puissance -4
0 x 2    "        2 ----- ! !   ! !   ----- 0 x 2    "        -3
1 x 2    "        1 ----- !   !   ----- 0 x 2    "        -2
0 x 2    "        0 -----          ----- 1 x 1    "        -1

```

Si nous additionnons le résultat des quatre premières lignes de gauche, qui constituent la partie entière de notre nombre, nous obtenons la valeur 10. Par ailleurs, si nous additionnons le résultat des 4 lignes de droite, qui constituent la partie décimale de notre nombre, nous obtenons la valeur 0.5. Nous constatons donc bien que cela correspond au nombre 10.5. Notons au passage l'importance de la virgule (représentée par un point) qui comme en base 10 délimite la partie entière de la partie décimale du nombre binaire.

Revenons maintenant à l'amstrad, et plus spécialement au codage d'un nombre flottant en mémoire, dans une instruction basic. Il faut bien comprendre qu'un ordinateur ne met pas de virgule réelle en mémoire, mais que sa position virtuelle est déterminée par un artifice, qui est en l'occurrence l'exposant. En effet l'exposant permet de savoir où se situe la virgule par rapport au dernier bit du nombre binaire, donc en partant de la gauche; dans notre exemple l'exposant serait égal à 4.

la représentation hexadécimale de 10.5 est :

```
< A8 00 00 00 04 >
```

< A8 00 00 00 > étant la mantisse du nombre, et < 04 > l'exposant.

Par ailleurs la représentation de 10.5 en mémoire de l'amstrad est:

< 1F 00 00 00 28 84 >

où < 1F > est fixe, < 00 00 00 28 > est la mantisse, et < 84 > l'exposant.

Nous voyons que l'exposant n'est autre que $04 + 80H$; nous retrouvons encore ce $80H$, qui, utilisé comme pour les noms de variables, permet d'indiquer au système qu'il est en présence d'un nombre flottant; en effet, la partie entière d'un nombre flottant ne pouvant dépasser 15 bits, le deuxième quartet de l'exposant est amplement suffisant. En ce qui concerne la mantisse on remarquera facilement qu'en ajoutant $80H$ au dernier octet, nous obtenons $A8H$; mais le résultat n'est pas tout à fait ce que l'on attend; toutefois si après ajout de $80H$, nous lisons la mantisse en partant de la droite, nous obtenons bien le résultat escompté.

Prenons un autre exemple pour bien fixer les idées.

Soit le nombre 567.4 qui en hexa donne < 8D D9 99 9A 0A >, avec < 8D D9 99 9A > comme mantisse, et < 0A > comme exposant.

Pour l'amstrad on aurait < 1F 9A 99 D9 0D 8A >, avec comme mantisse < 9A 99 D9 0D >, et < 8A > comme exposant. Après soustraction de $80H$ à $8A$ nous obtenons bien $0A$ comme exposant réel du nombre. Puis après addition de $80H$ à $0D$, et lecture en partant de la droite de la mantisse, nous obtenons bien < 8D D9 99 99 > comme mantisse réelle du nombre.

En annexe vous trouverez un court programme basic qui permet, à partir d'un nombre donné (entier ou flottant), de donner sa représentation binaire avec emplacement de la virgule.

d) nombres flottants négatifs:

De même que pour les nombres entiers négatifs, un octet d'une valeur hexa F5 est placé devant les octets adoptant le même principe que pour les nombres flottants positifs.

IDENTIFICATION DES VARIABLES

DANS LA "ZONE VARIABLE"

Nous avons vu que les variables étaient repérées par 2 octets. Plaçons nous juste après l'octet 00 de la dernière instruction basic; dans notre exemple de départ, en admettant que I est la première variable du programme basic, nous avons < 05 00 > pour valeur hexa des 2 octets de repérage. Positionnons nous 5 octets plus loin; nous nous trouvons juste après les 5 zéros de fin du programme basic; c'est ici que le système va trouver les éléments à traiter pour chaque variable. La fin de la première variable est délimitée par le début de la variable suivante, et ainsi de suite. Nous pouvons d'ores et déjà comprendre les premiers octets d'une variable dans cette zone; les suivants étant réservés au traitement même de la variable (calculs etc ...).

Comme dans le programme, les premiers octets identifient le nom de la variable, suivant le même principe; vient ensuite un octet permettant d'identifier le type de variable auquel nous avons affaire, mais les valeurs prises par cet octet sont différentes de celles prises par l'octet d'identification au niveau du programme.

<u>TYPE DE VARIABLE</u>	<u>! VALEUR</u>
Réelle	! 04
Chaine	! 02
Entière	! 01

Un octet 00 termine chaque variable, jouant un role de délimiteur.

CHAPITRE 4 - GESTION DU CLAVIER

I - GENERALITES:

Le clavier est le seul moyen permettant de dialoguer directement avec l'ordinateur; nous pouvons dénombrer un certain nombre de routines de saisie de caractères, de redefinition des touches, et d'autres, permettant une utilisation agréable de son ordinateur. Le clavier de l'amstrad est scindé en deux parties; d'une part le clavier classique "QWERTY", avec les fonctions DEL, CTRL, ESC, CAPS LOCK, TAB, SHIFT, et ENTER; d'autre part un petit clavier numérique dont toutes les touches peuvent être redéfinies comme étant des fonctions particulières (voir exemple d'application). Par ailleurs il peut être intéressant de redéfinir le clavier "QWERTY", par exemple en clavier "AZERTY"; les touches Q,W,E,R,T, et Y étant respectivement redéfinies en A,Z,E,R,T et Y.

De plus une saisie rapide de caractères peut également être envisagée pour des applications de gestions (saisie de données), ou tout simplement pour des jeux (de réflexes, etc ...). Les codes ASCII des touches du clavier "QWERTY" vont de 0 à 7FH, tandis que ceux du clavier numérique vont de 80H à 8CH.

II - ROUTINES UTILISEES:

Les routines que nous allons traiter dans ce chapitre, permettent la redéfinition des touches du clavier "QWERTY", quelqu'en soit le mode (NORMAL, SHIFT, ou CTRL); vous pourrez aussi créer des "chaînes d'extension" (redéfinition des touches de fonction); il vous sera aussi possible de gérer la vitesse de saisie des touches.

```

=====

! ROUTINE ! FONCTIONS: initialisation du clavier, soit:      !
!          ! - buffer et chaines d'extension                  !
! BB00 H ! - des touches (redéfinitions perdues)             !
! 47872 D ! - de la répétition - mise du clavier en minuscule !
!-----!-----!
!
!          PARAMETRES          ! INDICATEURS! REG. !
!
!          ENTREE              SORTIE      ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! néant                ! A = 0          ! oui      ! BC   !
!                      !                !          ! DE   !
!                      !                !          ! HL   !
!                      !                !          ! A    !
!                      !                !          !      !
!                      !                !          !      !
!                      !                !          !      !
!-----!-----!-----!-----!
! NOTES:                !
!                      !
=====

```

=====

! ROUTINE ! FONCTIONS: attend la frappe d'une touche au clavier. !

! ! !

! BB06 H ! !

! 47878 D ! !

! _____ ! _____ !

! PARAMETRES ! INDICATEURS! REG. !

! ENTREE SORTIE ! AFFECTES. ! AFF. !

! ----- ! ----- ! ----- ! ----- !

! néant ! A = code ASCII ! oui ! A !

! ! du ! ! !

! ! caractère ! ! !

! ! tape. ! ! !

! ! ! ! !

! ! ! ! !

! ----- !

! NOTES: !

! !

=====

```

=====

! ROUTINE ! FONCTIONS: va chercher un caractère du clavier      !
!
!      ! dans le buffer.                                         !
!
! BB09 H !                                                         !
!
! 47881 D !                                                         !
!
! _____! _____!
!
!              PARAMETRES              !INDICATEURS! REG. !
!
!      ENTREE              SORTIE      ! AFFECTES. ! AFF. !
!
! .....! .....! .....! .....!
!
! néant              ! A = code ASCII ! si trouve ! A      !
!
!              ! caractère ! C=1      !      !
!
!              ! si trouve. !          !      !
!
!              !          ! sinon      !      !
!
!              !          ! C=0        !      !
!
!              !          !          !      !
!
! .....!
!
! NOTES: la mise de caractère, dans le buffer, est faite par la !
!
! routine &BB0C.                                         !
!
=====

```

[illegible]

! ROUTINE ! FONCTIONS: place un caractère dans le buffer clavier. !

! ! ! !

! BBOC H !

! 47884 D !

! ! !

!	PARAMETRES	! INDICATEURS! REG. !
---	------------	-----------------------

!	ENTREE	SORTIE	! AFFECTES. !	AFF. !
---	--------	--------	---------------	--------

.....

```
! A = code ASCII du caractère!           !           !           !
```

```
!      a placer dans le buffer!      !      !      !
```

! clavier. ! ! ! !

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 1035 1036 1037 1038 1039 1040 1

! ! ! ! !

.....

! NOTES: le caractère sera appelé par la routine BB09 pour !

```
! affichage. !
```

[illegible]

```

=====

! ROUTINE ! FONCTIONS: crée une chaîne d'extension.      !

!           !                                           !

! BBOF  H  !                                           !

! 47887 D  !                                           !

! _____!_____!

!                               PARAMETRES                !INDICATEURS! REG. !

!           ENTREE                SORTIE      ! AFFECTES. ! AFF. !

! .....! .....! .....! .....!

! B = code de la touche      ! néant      ! si OK      ! BC  !

!   numérique redéfinie.    !           !   C=1      ! DE  !

! C = longueur de la chaîne !           ! sinon      ! HL  !

!   d'extension.            !           !   C=0      ! A   !

! HL = adresse de debut de  !           !           !     !

!   la chaîne.              !           !           !     !

! .....! .....! .....! .....!

! NOTES: la chaîne ne doit pas être trop longue (moins de 40c). !

!                                           !

=====

```



```

=====

! ROUTINE ! FONCTIONS: retire un caractère de la chaîne !
!          ! d'extension. !
! BB12 H !
! 47890 D !
! _____!
!
!          PARAMETRES          !INDICATEURS! REG. !
!          ENTREE              SORTIE    ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! A = code de la touche      ! si trouve      ! si trouve ! DE !
!   redéfinie.              ! A = caractère  !   C=1    ! A  !
! L = numéro du caractère    !                !          ! H  !
!   à enlever de la chaîne.!                ! sinon    !    !
!                            !                ! C=0      !    !
!                            !                !          !    !
!-----!
! NOTES: voir code des touches de fonction dans manuel !
! d'utilisation (80H a 8CH). !
=====

```

```

=====

! ROUTINE ! FONCTIONS: détermine une place pour logger les      !
!          ! chaînes d'extension.                                !
! BB15 H !                                                         !
! 47893 D !                                                         !
! _____! _____!
!
!          PARAMETRES          !INDICATEURS! REG. !
!
!          ENTREE              SORTIE      ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! DE = adresse du buffer      ! neant      ! si OK      !      !
!      pour les chaînes.      !          ! C=1        !      !
! HL = longueur du buffer     !          ! sinon      !      !
!          !          ! C=0        !      !
!          !          !          !      !
!          !          !          !      !
!-----!
! NOTES: ce buffer doit être suffisamment long (au moins 45 c)  !
!
=====

```

=====

! ROUTINE ! FONCTIONS: attend la frappe d'une touche du clavier !

! ! "QWERTY" ou d'une touche de fonction redfinie. !

! BB18 H ! !

! 47896 D ! !

! _____ !

! PARAMETRES !INDICATEURS! REG. !

! ENTREE SORTIE ! AFFECTES. ! AFF. !

!-----!-----!-----!-----!

! néant ! A = code ASCII ! C=1 ! non !

! ! de la touche. ! ! !

! ! ! ! !

! ! ! ! !

! ! ! ! !

! ! ! ! !

!-----!

! NOTES: !

! !

=====

```

=====

! ROUTINE ! FONCTIONS: va chercher le numéro d'une touche qui !
!         ! a été frappée. !
! BB1B H ! !
! 47899 D ! !
! _____! !

!               PARAMETRES               ! INDICATEURS! REG. !
!         ENTREE               SORTIE      ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! neant                ! si touche tapée ! si OK      ! A      !
! ex: SAIS CALL &BB1B   ! A = caractère ! C=1      !      !
! ---      JR  NC,SAIS  !               ! sinon     !      !
!               !               ! C=0      !      !
!               !               !          !      !
!               !               !          !      !
!-----!
! NOTES: cette routine n'attend pas la frappe d'une touche ; pour !
! simuler une attente , voir exemple ci-dessus. !
=====

```

```

=====

! ROUTINE ! FONCTIONS: teste si une touche a été pressée.      !
!          !                                                    !
! BB1E H !                                                    !
! 47902 D !                                                    !
! _____! _____!
!
!          PARAMETRES          !INDICATEURS! REG. !
!          ENTREE              SORTIE    ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! A = numéro touche a tester ! C = etat SHIFT ! touche OK ! HL !
! ex: TEST LD  A,66          ! ou CTRL.    ! Z=0    ! C    !
! ---      CALL &BB1E        ! C=0    -> normal ! sinon    ! A    !
!          JR  Z,TEST         ! C=20H -> shift ! Z=1    !      !
!          RET                ! C=80H -> ctrl  !      !      !
!          !                  !      !      !
!-----!
! NOTES: cette routine n'attend pas l'appui d'une touche; si on !
! veut attendre l'appui d'une touche, faire comme l'exemple.    !
=====

```

```

=====

! ROUTINE ! FONCTIONS: détermine si le clavier est en CAPS OFF, !
!          ! CAPS ON, ou CTRL CAPS. !
! BB21 H !
! 47905 D !
! _____!

!          PARAMETRES          !INDICATEURS! REG. !
!          ENTREE              SORTIE    ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! neant                ! HL = etat    ! oui      ! HL  !
!-----!          !          ! A      !
! si CAPS OFF -> L = 0 et H = 0          !          !      !
! si CAPS ON  -> L = 0 et H = FF          !          !      !
! si ctrl caps -> L = FF et H = 0          !          !      !
!          !          !          !      !
!-----!

! NOTES: CAPS ON -> minuscule / CAPS ON -> majuscule !
!          CTRL CAPS -> majuscule + caractères spéciaux. !

=====

```

```

=====
! ROUTINE ! FONCTIONS: redéfinition d'une touche du clavier      !
!          ! "QWERTY".                                           !
! BB27  H !                                                       !
! 47911 D !                                                       !
! _____!_____!
!
!          PARAMETRES          !INDICATEURS! REG. !
!
!          ENTREE              SORTIE      ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! A = numéro de la touche à ! néant          ! oui      ! HL  !
!   changer.                -----!          ! A    !
! B = code ASCII du nouveau ex: LD  A,27 (P) !          !
!   caractère.              --- LD  B,35 (#) !          !
!
!                          CALL &BB27  !          !
!
!                          !          !
!-----!
! NOTES:
!
=====

```

```

=====

! ROUTINE ! FONCTIONS: donne le code ASCII d'une touche donnée !
!
! par son numéro. !
!
! BB2A H !
!
! 47914 D !
!
! _____ !
!
! PARAMETRES ! INDICATEURS! REG. !
!
! ENTREE SORTIE ! AFFECTES. ! AFF. !
!
! .....! .....! .....! .....!
!
! A = numéro de la touche ! A = code ASCII ! non ! HL !
!
! (ex: numéro de la ! de la touche ! ! A !
!
! touche Z -> 71) ! dont le numero ! ! !
!
! ! a été indiqué ! ! !
!
! ! dans le reg. A ! ! !
!
! ! en entrée. ! ! !
!
! .....!
!
! NOTES: voir manuel utilisateur pour les numros de touche. !
!
! un numéro de touche faux donne un résultat insignifiant. !
=====

```


=====

! ROUTINE ! FONCTIONS: redéfinit le SHIFT d'une touche donnée en !

! ! un autre caractère. !

! BB2D H ! !

! 47917 D ! !

! _____! _____!

! PARAMETRES ! INDICATEURS! REG. !

! ENTREE SORTIE ! AFFECTES. ! AFF. !

!-----!-----!-----!-----!

! A = numéro de la touche à ! néant ! oui ! HL !

! redéfinir. ! ! ! A !

! B = code ASCII du nouveau ! ! ! !

! caractère. ! ! ! !

! ex: SHIFT (Z) redéfini en P! ! ! !

! --- ! ! ! !

!-----!

! NOTES: !

! !

=====

=====

```

=====

! ROUTINE ! FONCTIONS: donne le code ASCII du SHIFT d'une      !
!           ! touche donnée.                                     !
! BB30 H !                                                         !
! 47920 D !                                                         !
! _____!_____!
!
!           PARAMETRES           ! INDICATEURS! REG. !
!
!           ENTREE           SORTIE           ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! A = numéro de la touche      ! A = code ASCII ! oui      ! HL !
! donnée.                      ! du SHIFT de la !         ! A  !
!                               ! touche donnée. !         !   !
!                               !                 !         !   !
!                               !                 !         !   !
!                               !                 !         !   !
!-----!-----!-----!-----!
! NOTES: si le numéro de la touche donnée est invalide, le code !
! ASCII retourné est faux.                                     !
=====

```

=====

! ROUTINE ! FONCTIONS: redéfinit le CTRL d'une touche donnée en !

! ! un caractère donné. !

! BB33 H ! !

! 47923 D ! !

! _____ ! _____ !

! PARAMETRES ! INDICATEURS! REG. !

! ENTREE SORTIE ! AFFECTES. ! AFF. !

! - - - - - ! - - - - - ! - - - - - !

! A = numéro du CTRL de la ! néant ! oui ! HL !

! touche donnée. ! ! ! A !

! B = code ASCII du nouveau ! ! ! !

! caractère. ! ! ! !

! ex: CTRL (P) donnera X ! ! ! !

! --- ! ! ! !

! - - - - - !

! NOTES: !

! !

=====

```

=====

! ROUTINE ! FONCTIONS: donne le code ASCII du CTRL d'une      !
!          ! touche donnée.                                     !
! BB36 H !                                                    !
! 47926 D !                                                    !
! _____!_____!
!
!          PARAMETRES          ! INDICATEURS! REG. !
!
!          ENTREE              SORTIE      ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! A = numéro de la touche      ! A = code ASCII ! non      ! HL  !
! donnée.                      ! du CTRL de la !          ! A   !
!                              ! touche donnée. !          !    !
!                              !                !          !    !
!                              !                !          !    !
!                              !                !          !    !
!                              !                !          !    !
!-----!
! NOTES: si le numéro de touche donnée est invalide, le code . !
! ASCII retourné est incorrect.                                  !
=====

```

=====

! ROUTINE ! FONCTIONS: autorisation ou non de la répétition !

! ! d'une touche donnée, lorsque celle-ci sera appuyée !

! BB39 H ! de façon continue. !

! 47929 D ! !

! _____ !

! PARAMETRES ! INDICATEURS! REG. !

! ENTREE SORTIE ! AFFECTES. ! AFF. !

!-----!-----!-----!-----!

! A = numéro de la touche ! néant ! oui ! BC !

! donnée. ! ! ! HL !

! B = FFH -> répétition ! ! ! A !

! B = 0 -> non répétition ! ! ! !

! ! ! ! !

! ! ! ! !

!-----!

! NOTES: si le numéro de touche est incorrect, l'exécution de !

! la routine est sans conséquence. !

=====

```

=====

! ROUTINE ! FONCTIONS: demande si une touche donnée est soumise !
!
! ou non a une répétition, en cas d'appui en continu. !
! BB3C H !
! 47932 D !
! _____!
!
! PARAMETRES ! INDICATEURS! REG. !
!
! ENTREE SORTIE ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! A = numéro de la touche ! néant ! répétition! HL !
! donnée. ! Z=0 ! A !
! ! sinon ! !
! ! Z=1 ! !
! ! ! !
! ! ! !
! ! ! !
!-----!
! NOTES: si le numéro de touche est incorrect, le résultat !
! retourné est insignifiant. !
=====

```

```

=====
! ROUTINE ! FONCTIONS: change les délais de saisie d'un caractère.!
!
!      ! 1) délai de départ de saisie d'un caractère      !
!
! BB3F H ! 2) délai de répétition d'un caractère soumis    !
!
! 47935 D !      a une répétition.                          !
!
! _____! _____!
!
!      PARAMETRES      ! INDICATEURS! REG. !
!
!      ENTREE          SORTIE      ! AFFECTES. ! AFF. !
!
! -----! -----! -----! -----!
!
! H = délai de départ      ! néant      ! oui      ! A      !
!
! L = vitesse de répétition -----!      !      !
!
! ex: LD HL,3001 -> départ lent et vitesse de      !      !
!
! ---      ! !      répétition rapide.      !      !
!
!      H L      !      !      !
!
! nota: valeur 255 = 5 sec.      !      !      !
!
! -----!
!
! NOTES: a l'initialisation du clavier, le délai de départ est de !
!
! 02, et la vitesse de répétition de 1E.      !
!
=====

```

```

=====

! ROUTINE ! FONCTIONS: demande quel est le délai de départ, et !
!
!         ! quelle est la vitesse de répétition. !
!
! BB42 H !
!
! 47938 D !
!
! _____!
!
!               PARAMETRES                ! INDICATEURS! REG. !
!
!         ENTREE                SORTIE      ! AFFECTES. ! AFF. !
!
! .....! .....! .....! .....!
!
! néant                ! H = délai de    ! oui      ! A      !
!
!                     !   départ.    !          !        !
!
!                     ! L = vitesse de !          !        !
!
!                     !   répétition. !          !        !
!
!                     !          !          !        !
!
!                     !          !          !        !
!
! .....!
!
! NOTES: voir routine ci-dessus pour valeurs a l'initialisation. !
!
! _____!
=====

```


III - EXEMPLE D'UTILISATION:

Il est toujours ennuyeux, lors de la mise au point d'un programme, qu'il soit en basic ou en assembleur, de retaper toujours les mêmes ordres (load, save, run, call); aussi, puisque nous disposons de la possibilité de définir des fonctions au niveau des touches du clavier numérique, nous pouvons envisager un court programme en assembleur qui, chargé à la mise en marche de l'amstrad, donnerait à celui-ci les fonctions précédemment citées.

REPARTITION DES FONCTIONS:

touche 0 → fonction d'exécution d'un basic en mémoire: RUN

touche 1 → détermination d'un nom de fichier pour LOAD ou SAVE

touche 4 → LOAD d'un fichier suivant le nom donné ci-dessus

touche 7 → SAVE d'un fichier suivant le nom donné ci-dessus

touche 8 → détermination d'une adresse d'exécution pour ASS.

touche 9 → exécution d'un programme ASS. suivant adr. ci-dessus

FABRICATION D'UN AIDE - MEMOIRE EN CARTON :

Il peut être judicieux, et en tous cas efficace, de procéder a la fabrication d'un petit carton indiquant les fonctions des touches utilisées, et que l'on placerait sur le clavier numérique.

```
.....
!           ADR ASS           !
!           .....           !
!SAVE!7      8      9!CALL!
!      !           !      !
!LOAD!4           !      !
!      !           !      !
!NOM !1           !      !
!      !           !      !
!RUN !0           !      !
!           .....           !
! .....           !
.....
```

LISTING DU PROGRAMME ASSEMBLEUR:

```

1                ORG 7000H

2                LOAD $

3 7000 0680      LD  B,80H      ;TOUCHE 0

4 7002 0E04      LD  C,4

5 7004 210080    LD  HL,8000H  ;RUN

6 7007 CD0FBB    CALL 0BB0FH

7 700A 0681      LD  B,81H      ;TOUCHE 1

8 700C 0E1B      LD  C,27

9 700E 211080    LD  HL,8010H  ;NOM DE FICHIER

10 7011 CD0FBB   CALL 0BB0FH

11 7014 0684      LD  B,84H      ;TOUCHE 4

12 7016 0E0C      LD  C,12

13 7018 213080    LD  HL,8030H  ;LOAD NOM DE FICHIER

14 701B CD0FBB    CALL 0BB0FH

15 701E 0687      LD  B,87H      ;TOUCHE 7

16 7020 0E0C      LD  C,12

17 7022 214080    LD  HL,8040H  ;SAVE NOM DE FICHIER

18 7025 CD0FBB    CALL 0BB0FH

```

19 7028 0688	LD	B,88H	;TOUCHE 8
20 702A 0E1D	LD	C,29	
21 702C 215080	LD	HL,8050H	;ADR. EXEC. ASSEMBLEUR
22 702F CD0FBB	CALL	0BB0FH	
23 7032 0689	LD	B,89H	;TOUCHE 9
24 7034 0E0B	LD	C,11	
25 7036 217080	LD	HL,8070H	;CALL ADR. EXEC. ASS.
26 7039 CD0FBB	CALL	0BB0FH	
27 703C C9	RET		
28	END		

DUMP DES TEXTES UTILISES:

ADR !	TEXTES	! TOUCHES !
-------	--------	-------------

-----!

8000:	52 55 4E 0D 00 00 00 00 ! RUN..... !	0 !
-------	--------------------------------------	-----

!	!	!
---	---	---

8010:	43 4C 53 3A 49 4E 50 55 ! CLS:INPU !	1 !
-------	--------------------------------------	-----

8018:	54 22 4E 4F 4D 20 46 49 ! T"NOM FI !	!
-------	--------------------------------------	---

8020:	43 48 49 45 52 3A 22 3B ! CHIER:"; !	!
-------	--------------------------------------	---

8028:	4E 24 0D 00 00 00 00 00 ! N\$...... !	!
-------	---------------------------------------	---

!	!	!
---	---	---

8030:	43 4C 53 3A 4C 4F 41 44 ! CLS:LOAD !	4 !
-------	--------------------------------------	-----

8038:	20 4E 24 0D 00 00 00 00 ! N\$...... !	!
-------	---------------------------------------	---

!	!	!
---	---	---

8040:	43 4C 53 3A 53 41 56 45 ! CLS:SAVE !	7 !
-------	--------------------------------------	-----

8048:	20 4E 24 0D 00 00 00 00 ! N\$...... !	!
-------	---------------------------------------	---

!	!	!
---	---	---

8050:	43 4C 53 3A 49 4E 50 55 ! CLS:INPU !	8 !
-------	--------------------------------------	-----

8058:	54 22 41 44 52 2E 20 45 ! T"ADR. E !	!
-------	--------------------------------------	---

8060:	58 45 43 2E 20 41 53 53 ! XEC. ASS !	!
-------	--------------------------------------	---

8068:	3A 22 3B 45 0D 00 00 00 ! :";E.... !	!
-------	--------------------------------------	---

!	!	!
---	---	---

8070:	43 4C 53 3A 43 41 4C 4C ! CLS:CALL !	9 !
-------	--------------------------------------	-----

8078:	20 45 0D 00 00 00 00 00 ! E..... !	!
-------	------------------------------------	---

CHAPITRE 5 -

L'ECRAN TEXTE

I - GENERALITES:

Le CPC permet l'affichage de caractères selon trois modes. Le nombre de caractères affichés à l'écran ainsi que le nombre de couleurs utilisables parmi les 27 proposées par l'appareil sont en fonction du mode choisi:

- MODE 0: 25 lignes de 20 caractères - 16 couleurs utilisables
- MODE 1: 25 lignes de 40 caractères - 8 couleurs utilisables
- MODE 2: 25 lignes de 80 caractères - 2 couleurs utilisables

L'affichage et la gestion de l'écran peuvent s'effectuer de deux manières différentes:

- par utilisation d'une routine ROM située à l'adresse &C337 pour le CPC 464, à l'adresse &C380 pour le CPC 664 et à l'adresse &C37D pour le CPC 6128.

- par utilisation de routines RAM se trouvant dans la zone &BB4E - &BDF3.

II ROUTINE ROM :

Cette routine est utilisée par l'interpréteur BASIC pour afficher des caractères et les gérer. Elle permet au programmeur d'obtenir outre l'affichage, des effets très intéressants comme la création de fenêtres, la superposition de caractères, l'inversion vidéo, le déplacement du curseur, le choix des couleurs de l'encre du papier et du stylo ainsi que la création de symboles...

Son utilisation est simple puisqu'après son appel cette routine va charger dans HL l'adresse d'une série d'octets correspondant aux codes ASCII des caractères à afficher et aux codes des commandes de contrôle (à savoir : codes ASCII au numéro inférieur à 32 dont le tableau explicatif suit...).

CODES DE CONTROLE

NUMERO	!	EFFET	!	PARAMETRE
-----	!	-----	!	-----
	!		!	
* Code 0	!	aucun effet particulier.	!	
	!		!	
* Code 1	!	affiche le caractère donné par le	!	x param.
	!		!	
	!	paramètre qui suit.	!	de 0 à 255
	!		!	
* Code 2	!	efface le curseur de texte.	!	
	!		!	
* Code 3	!	rétablissement du curseur de texte.	!	
	!		!	
* Code 4	!	choix du mode écran.	!	de 0 à 2
	!		!	
* Code 5	!	positionne le caractère désigné par le	!	de 0 à 255
	!		!	
	!	paramètre sur le curseur graphique.	!	

	!	!
* Code 6	! rétablit l'écriture de texte stoppée par	!
	!	!
	! par l'envoi du code 21.	!
	!	!
* Code 7	! produit un beep sonore et vide les files	!
	!	!
	! d'attentes sonores.	!
	!	!
* Code 8	! décale le curseur d'une colonne vers la	!
	!	!
	! gauche.	!
	!	!
* Code 9	! décale le curseur d'une colonne vers la	!
	!	!
	! la droite.	!
	!	!
* Code 10	! décale le curseur d'une ligne vers le bas	!
	!	!
	! et le ramène à gauche de l'écran.	!

	!	!
* Code 11	! décale le curseur d'une ligne vers le haut !	
	!	!
	! sans le ramener à gauche de l'écran.	!
	!	!
* Code 12	! efface l'écran de texte et ramène le !	
	!	!
	! curseur en haut et à gauche de l'écran.	!
	!	!
* Code 13	! ramène le curseur au début de la ligne en !	
	!	!
	! cours.	!
	!	!
* Code 14	! fixe l'encre du papier.	! de 0 à 15
	!	!
* Code 15	! fixe l'encre du stylo.	! de 0 à 15

NUMERO	!	EFFET	!	PARAMETRES
-----	!	-----	!	-----
	!		!	
* Code 16	!	efface le caractère sous le curseur et met	!	
	!		!	
	!	à la place l'encre du papier.	!	
	!		!	
* Code 17	!	efface du bord gauche de la fenêtre à la	!	
	!		!	
	!	position du curseur et met à la place	!	
	!		!	
	!	l' encre du papier.	!	
	!		!	
* Code 18	!	efface la ligne de la fenêtre, du curseur	!	
	!		!	
	!	jusqu'au bord droit et met à la place	!	
	!		!	
	!	l' encre du papier.	!	
	!		!	
* Code 19	!	efface du début de la fenêtre jusqu'au	!	

	!	!
	! curseur et met à la place l'encre du !	
	!	!
	! papier.	!
	!	!
* Code 20	! efface la fenêtre, du curseur jusqu'à la !	
	!	!
	! fin et met à la place l'encre du papier. !	
	!	!
* Code 21	! bloque l'affichage sur l'écran texte !	
	!	!
	! jusqu'à l'envoi du code 6.	!
	!	!
* Code 22	! positionne en mode transparent ou opaque ! 1 ou 0	
	!	!
	! par envoi en paramètre de 1 ou 0.	!
	!	!
* Code 23	! permet le choix du mode graphique par ! 1 param.	
	!	!
	! l' envoi du paramètre suivant : 0 pour le ! de 0 à 3	

!	!
! mode NORMAL , 1 pour le mode XOR , 2 pour	!
!	!
! le mode AND et 3 pour le mode OR.	!
!	!
! (voir chapitre 7 pour explication).	!
!	!
* Code 24	! permet l'inversion vidéo par échange des
!	!
!	! couleurs de l'encre et du papier.
!	!
* Code 25	! permet la redéfinition d'un caractère par
!	! 9 param.
!	!
!	! l'envoi de 9 paramètres, le premier pour
!	! de 0 à 255
!	!
!	! le numéro du caractère et les 8 suivants
!	!
!	! pour chaque ligne de la nouvelle matrice.
!	!

NUMERO	!	EFFET	!	PARAMETRES
	!		!	
* Code 26	!	permet la création de fenêtre par l'envoi	!	2 param.
	!		!	
	!	de 4 paramètres fixant: le bord gauche,	!	de 1 à 80
	!		!	
	!	le bord droit, la ligne du haut et la	!	2 param.
	!		!	
	!	ligne du bas.	!	de 1 à 25
	!		!	
	!	Les bords gauche et haut doivent être <10.	!	
	!		!	
* Code 27	!	aucun effet particulier.	!	
	!		!	
* Code 28	!	fixe la ou les couleurs de l'encre par	!	1 param.
	!		!	
	!	l'envoi de 3 paramètres: 1 pour le numéro	!	de 0 à 15
	!		!	
	!	de l'encre, les 2 autres pour la ou les	!	2 param.
	!		!	

! couleurs (la deuxième, permettant le ! de 0 à 26

! !

! clignotement étant facultative). !

! !

* Code 29 ! fixe la ou les couleurs de la bordure par ! 2 param.

! !

! l'envoi de 2 paramètres: le premier pour ! de 0 à 26

! !

! une couleur fixe, le deuxième, facultatif !

! !

! pour obtenir un effet de clignotement. !

! !

* Code 30 ! positionne le curseur dans le coin haut!

! !

! et à gauche de la fenêtre. !

! !

* Code 31 ! positionne le curseur à un endroit de la ! 1 param.

! !

! fenêtre donné par les deux paramètres qui ! de 1 à 80

! !

! suivent.

! 1 param.

!

!

! Le code 30 doit être obligatoirement

! de 1 à 25

!

!

! exécuté avant utilisation du code 31.

!

=== EXEMPLE D'UTILISATION ===

Voici maintenant une démonstration de l'utilisation de la routine. Son emplacement se situera à partir de l'adresse &7000. Bien sur, vous pouvez, si vous le désirez, placer le programme à une toute autre adresse en fonction de vos besoins. La suite des codes ASCII utilisés par la routine après chargement dans le registre HL est située en &8000...

Vous trouverez ci-dessous: le listing du programme, le listing des codes et un schéma de la redéfinition du caractère 240.

=== LISTING DU PROGRAMME ===

```

                                ORG 7000H
                                LOAD 7000H
X:                                EQU 80C2H
Y:                                EQU 80C3H
AFF:                             EQU 0C337H      ;POUR LE CPC 464
A0:                              EQU 8000H      ;MODE 1
A1:                              EQU 8010H      ;FENETRE
A2:                              EQU 8020H      ;COULEURS
A3:                              EQU 8030H      ;RED. CARA.
A4:                              EQU 8040H      ;TEXTE 1
A5:                              EQU 8080H      ;CURSEUR BAS
A6:                              EQU 8090H      ;TEXTE 2
A7:                              EQU 80B0H      ;BIP
AB:                              EQU 80C0H      ;LOCATE
WAIT:                            EQU 7500H
ROUTA:                           EQU 7600H
SYMB:                            EQU 80C4H

7000 CD00B9      CALL 0B900H      ;ACCES ROM HAUTE
7003 210080      LD HL,A0
7006 CD37C3      CALL AFF
7009 211080      LD HL,A1
700C CD37C3      CALL AFF
700F 212080      LD HL,A2
7012 CD37C3      CALL AFF
7015 213080      LD HL,A3
7018 CD37C3      CALL AFF
701B 214080      LAF:          LD HL,A4
701E CD37C3      CALL AFF
7021 3E0F        LD A,0FH
7023 CD0075      LA:          CALL WAIT
7026 3D          DEC A
7027 20FA        JR NZ,LA
7029 0E0F        LD C,0FH
702B C5          LB:          PUSH BC
702C 21B080      LD HL,A5
702F CD37C3      CALL AFF
7032 C1          POP BC
7033 0D          DEC C
7034 20F5        JR NZ,LB
7036 219080      LD HL,A6
7039 CD37C3      CALL AFF
703C 3E09        LD A,9
703E CD0075      LC:          CALL WAIT
7041 3D          DEC A
7042 20FA        JR NZ,LC
7044 21B080      LD HL,A7

```

7047	CD37C3		CALL	AFF	
704A	3E01		LD	A,1	
704C	32C380		LD	(Y),A	
704F	061E		LD	B,30	
7051	0E00		LD	C,0	
7053	0C	LA1:	INC	C	
7054	79		LD	A,C	
7055	32C380		LD	(X),A	
7058	CD0076		CALL	ROUTA	
705B	10F6		DJNZ	LA1	
705D	21B080		LD	HL,A7	
7060	CD37C3		CALL	AFF	
7063	0608		LD	B,8	
7065	0E00		LD	C,0	
7067	0C	LB1:	INC	C	
7068	79		LD	A,C	
7069	32C380		LD	(Y),A	
706C	CD0076		CALL	ROUTA	
706F	10F6		DJNZ	LB1	
7071	21B080		LD	HL,A7	
7074	CD37C3		CALL	AFF	
7077	CD0075		CALL	WAIT	
707A	CD0075		CALL	WAIT	
707D	CD0075		CALL	WAIT	
7080	C9		RET		
			ORG	7500H	
			LOAD	\$	
7500	1E40		LD	E,40H	
7502	16FF	LW2:	LD	D,255	;WAIT 1/3 s
7504	FD210010	LW1:	LD	IY,1000H	
7508	15		DEC	D	
7509	20F9		JR	NZ,LW1	
750B	1D		DEC	E	
750C	20F4		JR	NZ,LW2	
750E	C9		RET		
			ORG	7600H	;ROUTA
			LOAD	\$	
7600	E5		PUSH	HL	
7601	C5		PUSH	BC	
7602	CD0075		CALL	WAIT	
7605	3EF0		LD	A,0F0H	
7607	32C480		LD	(SYMB),A	
760A	21C080		LD	HL,A8	
760D	CD37C3		CALL	AFF	
7610	CD0075		CALL	WAIT	
7613	3E20		LD	A,20H	
7615	32C480		LD	(SYMB),A	
7618	21C080		LD	HL,A8	
761B	CD37C3		CALL	AFF	
761E	C1		POP	BC	
761F	E1		POP	HL	
7620	C9		RET		
			END		

LISTING DES CODES

ZONE DES CODES		Nom de la zone	FONCTION
toute série de codes se termine par un 0. Pour un paramètre de valeur 0, entrer son code ASCII (30H).			
8000: 04 01 00 00 00 00 00 00 :	A0	passage en mode 1	
8008: 00 00 00 00 00 00 00 00 :			
8010: 0C 1A 05 22 05 0E 00 00 : ..."....	A1	création fenetre	
8018: 00 00 00 00 00 00 00 00 :			
8020: 1D 06 09 18 00 00 00 00 :	A2	couleurs	
8028: 00 00 00 00 00 00 00 00 :			
8030: 19 F0 18 3C 66 FF 66 24 : .p.<f f\$	A3	redef. du symb. 240	
8038: 5A 81 00 00 00 00 00 00 : Z.....			
8040: 0C 1F 05 02 4D 41 49 54 :MAIT	A4	affichage texte 1	
8048: 52 49 53 45 20 44 45 20 : RISE DE			
8050: 4C 27 41 4D 53 54 52 41 : L'AMSTRA			
8058: 44 1E 1F 0E 04 70 61 72 : D....Par			
8060: 1E 1F 09 06 52 4F 59 20 :ROY			
8068: 65 74 20 57 45 59 45 52 : et WEYER			
8070: 1E 1F 0E 08 31 39 38 35 :1985			
8078: 00 00 00 00 00 00 00 00 :			
8080: 0A 00 00 00 00 00 00 00 :	A5	remontée texte 1	
8088: 00 00 00 00 00 00 00 00 :			
8090: 1E 1F 05 06 45 58 45 4D :EXEM	A6	affichage texte 2	
8098: 50 4C 45 20 44 45 20 44 : PLE DE D			
80A0: 45 46 49 4C 45 4D 45 4E : EFILEMEN			
80A8: 54 00 00 00 00 00 00 00 : T.....			
80B0: 07 00 00 00 00 00 00 00 :	A7	sonnette	
80B8: 00 00 00 00 00 00 00 00 :			
80C0: 1E 1F 1E 08 20 00 00 00 :	A8	pos. x,y et affichage symb. 240	
80C8: 00 00 00 00 00 00 00 00 :			
80D0: 00 00 00 00 00 00 00 00 :			

REDEFINITION DU SYMBOLE 240

8	4	2	1	8	4	2	1	
								18
								3C
								66
								FF
								66
								24
								5A
								81

III ROUTINES UTILISEES:

Ces routines seront décrites dans l'ordre de leur placement dans la RAM. De ce fait certaines routines utilisées pour l'impression du texte peuvent être traitées dans les chapitres suivants. C'est par exemple le cas de la routine permettant le choix du mode texte, placée en &BC0E, qui sera détaillée dans le chapitre concernant la mémoire écran. Trois exemples d'utilisation suivront le détail des routines. Pour chacune d'entre elles, il sera précisé s'il y a lieu de se référer à un de ces exemples....

```

=====

! ROUTINE ! FONCTIONS: Initialisation de l'écran de texte avec !
!           ! mise des couleurs 0 pour le papier, 1 pour le stylo, !
! BB4E H ! suppression des fenêtres, positionnement du curseur !
! 47950 D ! en haut et à gauche de l'écran, choix du canal 0... !
! _____!
!
!           PARAMETRES           ! INDICATEURS! REG. !
!
!           ENTREE           SORTIE           ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
!
!           !           ! Z=1 ; N=1 ! A !
!
!           !           !           ! BC !
!
!           !           ! reste à 0 ! DE !
!
!           !           !           ! HL !
!
!           !           !           !
!
!           !           !           !
!-----!
! NOTES: Cette routine n'efface pas l'écran... !
!
!           Voir exemples 1, 2 et 3. !
=====

```

```

=====
! ROUTINE ! FONCTIONS: Rétablissement de l'affichage de texte si !
!
! celui-ci était interdit par appel de la routine BB57. !
!
! BB54 H !
!
! 47956 D !
!
! _____ !
!
!          PARAMETRES          ! INDICATEURS! REG. !
!
!          ENTREE              SORTIE    ! AFFECTES. ! AFF. !
!
! .....! .....! .....! .....!
!
!          !          ! Z=1 ; 0=1 ! A !
!
!          !          !          !    !
!
!          !          ! reste à 0 !    !
!
!          !          !          !    !
!
!          !          !          !    !
!
!          !          !          !    !
!
! .....!
!
! NOTES:
!
!
=====

```

```

=====

! ROUTINE ! FONCTIONS: Bloque l'affichage des caractères appelés !
!           ! par les routines BB5A ou BB5D.           !
! BB57 H !                                           !
! 47959 D !                                           !
! _____!_____!
!
!           PARAMETRES           ! INDICATEURS! REG. !
!
!           ENTREE           SORTIE ! AFFECTES. ! AFF. !
!
! .....! .....! .....! .....!
!
!           !           ! Z=1 ; 0=1 ! A !
!
!           !           !           !   !
!
!           !           ! reste à 0 !   !
!
!           !           !           !   !
!
!           !           !           !   !
!
!           !           !           !   !
! .....!
!
! NOTES: Cette routine bloque également l'affichage du curseur. !
! Le rétablissement de l'affichage se fait par la routine BB54. !
=====

```



```

=====
! ROUTINE ! FONCTIONS: Affichage du caractère dont le code ASCII !
!
! est placé dans le registre A. Cette routine accepte !
! BB5A H ! également les codes de controle et leur obéi sans les !
! 47962 D ! afficher. !
! _____ !
!
! PARAMETRES !INDICATEURS! REG. !
!
! ENTREE SORTIE ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! A contient le code ASCII ! ! !
! du caractère à afficher ou ! ! !
! le code de controle. ! ! !
! ! ! !
! ! ! !
! ! ! !
! ! ! !
!-----!
! NOTES: Si le code utilisé nécessite des paramètres, ils seront !
! placés dans le registre A à la suite du code. Voir exemples 1,3.!
=====

```

```

=====

! ROUTINE ! FONCTIONS: Affiche un caractère à l'écran, dans la !
!
! fenêtr e en cours à la position actuelle du curseur. !
!
! BB5D H !
!
! 47965 D !
!
! _____!
!
! PARAMETRES !INDICATEURS! REG. !
!
! ENTREE SORTIE ! AFFECTES. ! AFF. !
!
!-----!-----!-----!-----!
!
! A contient le code ASCII ! tous les ! A !
!
! du caractère à afficher. ! ! BC !
!
! ! !indicateurs! DE !
!
! ! ! HL !
!
! ! mis à 0 ! !
!
! ! ! !
!
!-----!
!
! NOTES: Après appel de la routine, le curseur est déplacé d'une !
!
! colonne vers la droite. Voir exemple 2. !
!
=====

```

```

=====
! ROUTINE ! FONCTIONS: Lecture d'un caractère de l'écran à la !
!          ! position actuelle du curseur.                      !
! BB60 H !                                                         !
! 47968 D !                                                         !
! _____!_____!
!
!          PARAMETRES          !INDICATEURS! REG. !
!
!          ENTREE              SORTIE    ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
!
!          ! si un caractère ! -> C=1    ! HL !
!
!          ! est reconnu, A !          !      !
!
!          ! contient son   !          !      !
!
!          ! code ASCII.    !          !      !
!
!          ! sinon A=0.      ! -> C=0    !      !
!
!          !                !          !      !
!-----!
! NOTES: Le curseur doit etre à l'intérieur de la fenetre en cours!
!
=====

```

```

=====

! ROUTINE ! FONCTIONS: Possibilité ou interdiction d'affichage !

!           ! graphique de caractères. L'affichage se fait alors !

! BB63 H ! à la position du curseur graphique en sachant que !

! 47971 D ! celui-ci se trouve au coin haut à gauche du caractère !

! _____!

!           PARAMETRES           ! INDICATEURS! REG. !

!           ENTREE           SORTIE ! AFFECTES. ! AFF. !

!-----!-----!-----!-----!

! affichage possible: A<>0 !           !           !

!           !           !           !

! affichage interdit: A=0 !           !           !

!           !           !           !

!           !           !           !

!           !           !           !

!-----!

! NOTES: Après appel de cette routine, les codes de controles ne !

! seront plus obéis mais affichés par les routines BB5A et BB5D. !

=====

```

=====

! ROUTINE ! FONCTIONS: Création d'une fenêtre. Les bords internes !

! ! de la fenêtre sont données par rapport à une origine !

! BB66 H ! située en haut et à gauche de l'écran repérée par les !

! 47974 D ! coordonnées 0,0. !

! _____ !

! PARAMETRES ! INDICATEURS! REG. !

! ENTREE SORTIE ! AFFECTES. ! AFF. !

!-----!-----!-----!-----!

! H -> colonne gauche ! ! tous les ! A !

! D -> colonne droite ! ! ! BC !

! L -> ligne du haut ! ! indicateurs! !

! E -> ligne du bas ! ! ! !

! Si les paramètres sont trop! ! mis à 0 ! !

! grands, ils sont réduits. ! ! ! !

!-----!

! NOTES: Le curseur est placé en haut et à gauche de la fenêtre. !

! Cette routine n'efface pas le contenu de la fenêtre. Voir ex 1. !

=====

```

=====
! ROUTINE ! FONCTIONS: Indication des dimensions de la fenêtre en !
!          ! cours. Les limites de la fenêtre sont données en !
! BB69 H ! utilisant le même système de coordonnées que la !
! 47977 D ! routine précédente. !

```

```

! _____!

```

```

!          PARAMETRES          ! INDICATEURS! REG. !

```

```

!          ENTREE              SORTIE      ! AFFECTES. ! AFF. !

```

```

! .....! .....! .....! .....!

```

```

!          ! H colonne gauche! fenetre ! A !

```

```

!          ! L ligne haut    ! = écran ! BC !

```

```

!          ! D colonne droite! -> C=0  !    !

```

```

!          ! E ligne bas     ! fenetre !    !

```

```

!          !                  ! < écran !    !

```

```

!          !                  ! -> C=1  !    !

```

```

! .....!

```

```

! NOTES: !

```

```

!          !

```

```

=====

```

```

=====

! ROUTINE ! FONCTIONS: Effaçage de la fenêtre en cours.      !

!           !                                                    !

! BB6C  H !                                                    !

! 47980 D !                                                    !

! _____! _____!

!                                     PARAMETRES                !INDICATEURS! REG. !

!           ENTREE                SORTIE      ! AFFECTES. ! AFF. !

! -----! -----! -----! -----!

!           !                ! tous les ! A !

!           !                !          ! BC !

!           !                !indicateurs! DE !

!           !                !          ! HL !

!           !                ! mis à 0. !      !

!           !                !          !      !

! -----! -----! -----! -----!

! NOTES: Le curseur est placé en haut et à gauche de la fenêtre. !

! Voir exemples 1, 2 et 3.                                     !

=====

```

```

=====

! ROUTINE ! FONCTIONS: Positionnement horizontal du curseur.      !
!          !                                                         !
! BB6F  H !                                                         !
! 47983 D !                                                         !
! _____!_____!

!          PARAMETRES          ! INDICATEURS! REG. !
!          ENTREE              SORTIE    ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
!          !                  ! tous les. ! A  !
! A -> colonne du curseur  !          !          ! HL  !
!          !                  ! indicateurs!      !
!          !                  !          !      !
!          !                  ! mis à 0  !      !
!          !                  !          !      !
!-----!

! NOTES: Le curseur peut etre placé à l'extérieur de la fenêtre !
! en cours mais attention si un affichage doit avoir lieu.      !
=====

```


=====

! ROUTINE ! FONCTIONS: Positionnement vertical du curseur. !

! ! !

! BB72 H ! !

! 47986 D ! !

! _____ ! _____ !

! PARAMETRES ! INDICATEURS! REG. !

! ENTREE SORTIE ! AFFECTES. ! AFF. !

!!!!!

! A -> ligne du curseur ! tous les ! A !

! ! ! HL !

! ! !indicateurs! !

! ! ! ! !

! ! ! mis à 0. ! !

! ! ! ! !

!!

! NOTES: Le curseur peut être placé en dehors de la fenêtre en !

! cours mais attention en cas d'affichage de caractère. !

=====

```

=====
! ROUTINE ! FONCTIONS: Positionnement du curseur. !
! ! !
! BB75 H ! !
! 47989 D ! !
! _____ !
!
! PARAMETRES ! INDICATEURS! REG. !
!
! ENTREE SORTIE ! AFFECTES. ! AFF. !
! .....! .....! .....! .....!
! H -> colonne du curseur. ! ! tous les ! A !
! L -> ligne du curseur. ! ! ! HL !
! ! ! indicateurs! !
! ! ! ! !
! ! ! mis à 0. ! !
! ! ! ! !
! .....!
!
! NOTES: Le curseur peut être placé en dehors de la fenêtre en !
! cours mais attention en cas d'affichage de caractère. Voir ex 2 !
=====

```

```

=====
! ROUTINE ! FONCTIONS: Indication des coordonnées du curseur de !
!           ! la fenêtre en cours.                               !
! BB78 H !                                                         !
! 47992 D !                                                         !
! _____! _____!
!
!           PARAMETRES                ! INDICATEURS! REG. !
!
!           ENTREE                     SORTIE    ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
!
!           ! H colon. curseur! C=1    !      !
!
!           ! L ligne curseur !      !      !
!
!           !                   ! reste à 0 !      !
!
!           ! A compteur du    !      !      !
!
!           ! scrolling        !      !      !
!
!           !                   !      !      !
!-----!
! NOTES: A indique par comparaison avec une précédente valeur s'il
! y a eu un scrolling de la fenêtre en cours.
=====

```

```

=====
! ROUTINE ! FONCTIONS: Rétablissement de l'affichage du curseur !
!
!      ! dans la fenêtre en cours si celui-ci était interdit !
!
! BB7B H ! par appel de la routine BB7E.
!
! 47995 D !
!
! _____ !

```

```

!
!          PARAMETRES
!
!          ENTREE          SORTIE          ! INDICATEURS! REG. !
!
!          .....|.....|.....|.....|
!
!          !          ! tous les ! A !
!
!          !          !          !
!
!          !          ! indicateurs!
!
!          !          !          !
!
!          !          ! mis à 0. !
!
!          !          !          !
!
! .....|

```

```

! NOTES:
!
!
=====

```

```

=====

! ROUTINE ! FONCTIONS: Interdit l'affichage du bloc curseur dans !
!
! la fenêtre en cours. !
!
! BB7E H !
!
! 47998 D !
!
! _____!
!
! PARAMETRES ! INDICATEURS! REG. !
!
! ENTREE SORTIE ! AFFECTES. ! AFF. !
!
!-----!-----!-----!-----!
!
! ! ! O=1 ! A !
!
! ! ! ! !
!
! ! ! reste à 0 ! !
!
! ! ! ! !
!
! ! ! ! !
!
! ! ! ! !
!
!-----!
!
! NOTES: Le rétablissement de l'affichage du bloc curseur se fait !
! l'appel de la routine BB7B. !
=====

```

```

=====

! ROUTINE ! FONCTIONS: Indique si une position donnée du curseur !
!
! se trouve à l'intérieur de la fenêtre en cours et si !
! 8887 H ! l'impression d'un caractère nécessite un scrolling de !
! 48007 D ! la fenêtre ou un déplacement du curseur. !
! _____!
!
! PARAMETRES ! INDICATEURS! REG. !
!
! ENTREE SORTIE ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! H -> colonne du curseur ! H colon curseur ! 1. C=1 ! A !
! L -> ligne du curseur ! L ligne curseur ! ! !
!-----! 2. C=0 ! 1.B !
! 1. curs. à droite, à gauche ou dans fenêtre ! ! !
! 2. curs. sous la fenêtre(scroll haut -> B=FF)! 3. C=0 ! !
! 3. curs. au dessus fen. (scroll bas -> B=00)! ! !
!-----!
! NOTES: En sortie les registres H et L indiquent les nouvelles !
! coordonnées du curseur si celui-ci a été déplacé. !
=====

```

```

=====

! ROUTINE ! FONCTIONS: Affichage d'un bloc  curseur à la position !
!
!      ! actuelle du curseur  dans la fenêtre  en cours. Cette !
!
! BB8A H ! routine permet l'affichage de plusieurs curseurs.    !
!
! 48010 D !
!
! _____!
!
!              PARAMETRES              ! INDICATEURS! REG. !
!
!      ENTREE              SORTIE      ! AFFECTES. ! AFF. !
!
!-----!-----!-----!-----!
!
!              !              ! Z=1 ; N=1 ! A  !
!
!              !              !              !      !
!
!              !              ! reste à 0 !      !
!
!              !              !              !      !
!
!              !              !              !      !
!
!              !              !              !      !
!
!-----!
!
! NOTES: Si la position du curseur n'est pas  à L'intérieur de la !
!
! fenêtre celui-ci y est placé automatiquement.                  !
!
=====

```

```

=====

! ROUTINE ! FONCTIONS: Retire un bloc curseur à la position en !
!          ! cours du curseur.                                !
! BB80 H !                                                    !
! 48013 D !                                                    !
! _____!_____!
!
!          PARAMETRES          ! INDICATEURS! REG. !
!
!          ENTREE              SORTIE    ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
!
!          !                  ! Z=1 ; N=1 ! A !
!
!          !                  !          !    !
!
!          !                  ! reste à 0 !    !
!
!          !                  !          !    !
!
!          !                  !          !    !
!
!          !                  !          !    !
!-----!-----!-----!-----!
! NOTES: Cette routine a l'effet inverse de la routine précédente.!
!
=====

```


=====

! ROUTINE ! FONCTIONS: Fixe l'encre du stylo. (Voir exemple 1) !

! ! !

! BB90 H ! !

! 48016 D ! !

! _____ !

! PARAMETRES !INDICATEURS! REG. !

! ENTREE SORTIE ! AFFECTES. ! AFF. !

!-----!-----!-----!-----!

! A -> numéro de l'encre. ! tous les ! A !

! ! ! HL !

! !indicateurs! !

! ! ! !

! ! mis à 0 ! !

! ! ! !

!-----!

! NOTES: Si la valeur placée en A est illégale, le système calcule!

! et utilise le modulo 2,4 ou 16 de la valeur selon le mode choisi!

=====

=====

! ROUTINE ! FONCTIONS: Renseigne sur l'encre d'écriture en cours. !

! ! !

! BB93 H ! !

! 48019 D ! !

! _____ !

! PARAMETRES ! INDICATEURS ! REG. !

! ENTREE SORTIE ! AFFECTES. ! AFF. !

! ! ! ! !

! ! A = numéro de ! Z=1 ; N=1 ! !

! ! ! ! !

! ! l'encre en cours ! reste à 0 ! !

! ! ! ! !

! ! ! ! !

! ! ! ! !

! !

! NOTES: !

! !

=====

=====

! ROUTINE ! FONCTIONS: Fixe l'encre du papier. !

! ! !

! BB96 H ! !

! 48022 D ! !

! _____ ! _____ !

! PARAMETRES !INDICATEURS! REG. !

! ENTREE SORTIE ! AFFECTES. ! AFF. !

!-----!-----!-----!-----!

! A -> numéro de l'encre du ! tous les ! A !

! papier ! HL !

! !indicateurs! !

! ! !

! ! mis à 0 ! !

! ! !

!-----!

! NOTES: Comme la routine BB90 le système utilise le modulo 2, 4 !

! ou 16 du registre A en cas de valeur erronée. Voir exemple 1. !

=====

```

=====

! ROUTINE ! FONCTIONS: Renseigne sur l'encre utilisée pour le !
!          ! papier.                                           !
! BB99 H !                                                    !
! 48025 D !                                                    !
! _____!_____!
!
!          PARAMETRES          !INDICATEURS! REG. !
!
!          ENTREE              SORTIE      ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
!
!          ! A = numéro de   ! Z=1 ; N=1 !      !
!
!          !l'encre du papier!      !      !
!
!          !                  ! reste à 0 !      !
!
!          !                  !          !      !
!
!          !                  !          !      !
!
!          !                  !          !      !
!-----!
! NOTES:                                     !
!
=====

```

=====

! ROUTINE ! FONCTIONS: Vidéo inverse par échange des couleurs de !

! l'encre de l'écriture et du papier. !

! BB9C H !

! 48028 D !

! _____ !

! PARAMETRES ! INDICATEURS! REG. !

! ENTREE SORTIE ! AFFECTES. ! AFF. !

! - - - - - ! - - - - - ! - - - - - ! - - - - - !

! ! ! A !

! ! ! HL !

! ! ! !

! ! ! !

! ! ! !

! ! ! !

! - - - - - !

! NOTES: !

! !

=====

```

=====

! ROUTINE ! FONCTIONS: Choix du mode d'écriture transparent ou !
!
!      ! opaque.Cette routine permet de superposer plusieurs !
! BB9F H ! caractères dans le cas du mode transparent.      !
! 48031 D !
!
! _____!
!
!          PARAMETRES          !INDICATEURS! REG. !
!
!      ENTREE          SORTIE      ! AFFECTES. ! AFF. !
!
!-----!-----!-----!-----!
!
! A = 0 -> mode opaque          !          ! opaque:  ! A  !
!
!          !          ! Z=1 ; 0=1 ! HL !
!
! A <> 0 -> mode transparent !          !          !
!
!          !          ! transpar: !          !
!
!          !          ! tout à 0 !          !
!
!          !          !          !          !
!
!-----!
!
! NOTES: Le mode graphique est systématiquement opaque. Voir ex 1 !
!
!          !
=====

```

=====

! ROUTINE ! FONCTIONS: Renseigne si le mode d'écriture en cours !

! ! est opaque ou transparent. !

! BBA2 H !

! 48034 D !

! _____ !

! PARAMETRES ! INDICATEURS ! REG. !

! ENTREE SORTIE ! AFFECTES. ! AFF. !

! ! ! ! !

! opaque: A=0 !-> Z=1;0=1 ! DE !

! ! ! HL !

! transp: A<>0 !-> bit 3=1 ! !

! ! reste à 0 ! !

! ! ! !

! ! ! !

! !

! NOTES: Le mode graphique est systématiquement opaque. !

! !

=====

Les quatre routines suivantes appellent quelques commentaires de par leur importance. En effet, elles permettent la redéfinition de caractères. Si pour le BASIC celle-ci s'effectue simplement par l'utilisation de l'ordre SYMBOL, la démarche à suivre dans le cadre de programmes en langage machine est plus délicate...

Les 256 caractères utilisables sont stockés dans la ROM basse sous la forme de matrices de huit octets indiquant chacun l'état de chacune des huit lignes formant un caractère à l'écran. Pour modifier la forme d'un caractère il faut donc changer sa matrice mais il est impossible à l'utilisateur d'écrire et ainsi de changer quoique ce soit dans la ROM. Il faudra alors par l'intermédiaire de la routine BBABH recopier dans la RAM les matrices des caractères depuis le code ASCII voulu jusqu'au caractère de code 255.

L'utilisateur indiquera également la zone où se fera le stockage des matrices qui se situera entre les adresses 4000H et AB79H. Pour effectuer alors une redéfinition, il suffit à l'utilisateur soit d'appeler la routine BBA8H en indiquant le code ASCII du caractère à modifier ainsi que la zone où se trouve la nouvelle matrice, soit de changer les matrices à leurs adresses en cours dans la RAM.


```

=====

! ROUTINE ! FONCTIONS: Indication de l'adresse de la matrice d'un !
!          ! caractère. Si le caractère fait partie des caractères !
! BBA5 H ! copiés en RAM alors HL contient une adresse RAM !
! 48037 D ! sinon HL contient une adresse ROM. !
! _____ !
!
!          PARAMETRES          ! INDICATEURS! REG. !
!
!          ENTREE              SORTIE    ! AFFECTES. ! AFF. !
! -----!-----!-----!-----!
! A -> code ASCII du caract. ! HL -> adr. matr ! * redef ! A !
!
!      dont l'adresse de la !          ! possible !      !
!
!      matrice est cherchée. !          ! C=1      !      !
!
!          !          ! * redef !      !
!
!          !          ! impossible! !      !
!
!          !          ! C=0      !      !
!
! -----!
! NOTES: !
!
!
=====

```

```

=====
! ROUTINE ! FONCTIONS: Changement de la  matrice d'un caractère à !
!           ! condition qu'il ait été recopié dans la RAM sinon la !
! BBA8 H ! routine n'a pas d'effet. Le caractère sera toujours !
! 48040 D ! affiché sous sa nouvelle forme sauf réinitialisation. !
! _____! _____!
!
!           PARAMETRES           !INDICATEURS! REG. !
!
!           ENTREE           SORTIE   ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! A -> code ASCII du caract. !           ! redef.   ! A   !
!           à redéfinir.      !           ! possible ! BC  !
!           !                 ! -> C=1    ! DE  !
! HL -> adresse de stockage !           !           ! HL  !
!           de la nouvelle matr. !           ! sinon C=0 !           !
!           !                 !           !           !           !
!-----!-----!-----!-----!
! NOTES:                                     !
!                                           !
=====

```

```

=====
! ROUTINE ! FONCTIONS: Réécriture des matrices de caractères dans !
!
!      ! une zone RAM définie par l'utilisateur. Toutes les !
! BBAB H ! matrices depuis le caractère indiqué jusqu' à celui !
! 48043 D ! de code 256 sont stockées. Prévoir la place !
! _____ !
!
!              PARAMETRES              ! INDICATEURS! REG. !
!
!      ENTREE              SORTIE      ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! DE -> code ASCII du premier! 1: existence      ! 1: C=1      ! BC !
!
!      caractère à recopier.! d'un ancien stoc!      ! DE !
!
!              ! A= code 1er cara!      !      !
!
! HL -> adresse de la zone ! HL= zone stocka.! 2: C=0      ! 2:A !
!
!      de stockage des mat. ! 2: pas d'ancien !      ! HL !
!
!              !      stockage      !      !
!-----!
! NOTES: L'adresse de zone doit être comprise entre 4000H et AB79H!
! Cette zone peut être recomplétée si besoin. Voir exemple 3.      !
=====

```

```

=====

! ROUTINE ! FONCTIONS: Indication de l'existence et de l'adresse !
!         ! d'une zone de stockage des matrices de caractères.   !
! BBAE H !                                                         !
! 48046 D !                                                         !
! _____!_____!
!
!               PARAMETRES                !INDICATEURS! REG. !
!
!      ENTREE                SORTIE      ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
!
!               ! 1: zone définie ! 1: C=1   ! 1:   !
!               ! A= 1er caractère!         !     !
!               ! HL= adresse zone!         !     !
!               !                 !         !
!               ! 2: pas de zone  ! 2: C=0   ! 2:A  !
!               ! rien en sortie  !         !   HL !
!-----!
! NOTES:                                     !
!
=====

```

```

=====
! ROUTINE ! FONCTIONS: Choix du canal ou de la fenêtre en cours. !
!           !
! BBB4 H !
! 48052 D !
! _____ !
!
!           PARAMETRES           ! INDICATEURS! REG. !
!           ENTREE               SORTIE  ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! A -> canal choisi              !           ! HL !
!
!
!
!
!
!
!
!-----!
! NOTES: Si le numéro du canal est supérieur à 8 le système prend !
! la valeur du registre A modulo 8. Voir exemple 1.           !
=====

```

```

=====
! ROUTINE ! FONCTIONS: Inversion de l'état de 2 canaux ou fenêtres!
!
! L'inversion touche les couleurs d'encre du stylo et du
! BBB7 H ! papier, le mode d'écriture, les bornes des fenêtres et!
! 48055 D ! la position du curseur.
!
! _____!
!
! PARAMETRES ! INDICATEURS! REG. !
!
! ENTREE SORTIE ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! B -> numéro premier canal ! S=1 ! A !
! ! ! DE !
! C -> numéro second canal ! bit 5 = 0 ! HL !
! ! ! !
! ! ! reste à 0 !
! ! ! !
!-----!
! NOTES: Si les numéros des canaux sont illégaux (>8), le système !
! utilise leurs modulus 8. Voir exemple 1.
!
=====

```

=== EXEMPLES D'UTILISATION ===

Voici trois exemples d'utilisation. Le premier créera deux fenêtres puis fixera les couleurs de l'encre du papier et du stylo de chacune ainsi que celle de l'écran. Chaque fenêtre portera son numéro. Un court texte indiquera que l'appui d'une touche inversera l'état des deux fenêtres... Le deuxième exemple mettra en évidence l'utilisation du mode d'écriture transparent. Enfin, le troisième montrera les possibilités de redéfinition de caractères...

```
                                ORG    7000H

                                LOAD   $

TEXT1:    EQU    7200H

TEXT2:    EQU    7210H

TEXT3:    EQU    7220H

7000  CD4EBB          CALL  0BB4EH    ;INITALISATION

7003  3E02           LD     A,2

7005  CD96BB          CALL  0BB96H    ;ENCRE PAPIER

7008  3E00           LD     A,0

700A  CD90BB          CALL  0BB90H    ;ENCRE STYLO

700D  CD06BB          CALL  0BB6CH    ;CLS
```

7010	3E01	LD	A,1	
7012	CDB4BB	CALL	0BBB4H	;CANAL 1
7015	2601	LD	H,1	
7017	160B	LD	D,11	
7019	2E01	LD	L,1	
701B	1E0A	LD	E,10	
701D	CD66BB	CALL	0BB66H	;CREATION FENETRE 1
7020	3E00	LD	A,0	
7022	CD96BB	CALL	0BB96H	;ENCRE PAPIER FEN 1
7025	3E01	LD	A,1	
7027	CD90BB	CALL	0BB90H	;ENCRE STYLO FEN 1
702A	CD06BB	CALL	0BB6CH	;CLS FENETRE 1
702D	3E02	LD	A,2	
702F	CDB4BB	CALL	0BBB4H	;CANAL 2
7032	2615	LD	H,21	
7034	161F	LD	D,31	
7036	2E0F	LD	L,15	
7038	1E14	LD	E,20	
703A	CD66BB	CALL	0BB66H	;CREATION FENETRE 2
703D	3E03	LD	A,3	

703F	CD96BB		CALL	0BB96H	;ENCRE PAPIER FEN 2
7042	3E02		LD	A,2	
7044	CD90BB		CALL	0BB90H	;ENCRE STYLO FEN 2
7047	CD6CBB		CALL	0BB6CH	;CLS FENETRE 2
704A	3E01	DEBUT:	LD	A,1	
704C	CDB4BB		CALL	0BBB4H	;CANAL 1
704F	CD6CBB		CALL	0BB6CH	;CLS FENETRE 1
7052	210072		LD	HL,TEXT1	
7055	7E	L1:	LD	A,(HL)	
7056	CD5ABB		CALL	0BB5AH	;TEXTE 1
7059	23		INC	HL	
705A	3E20		LD	A,32	
705C	BE		CP	(HL)	
705D	20F6		JR	NZ,L1	
705F	3E02		LD	A,2	
7061	CDB4BB		CALL	0BBB4H	;CANAL 2
7064	CD6CBB		CALL	0BB6CH	;CLS FENETRE 2
7067	211072		LD	HL,TEXT2	
706A	7E	L2:	LD	A,(HL)	
706B	CD5ABB		CALL	0BB5AH	;TEXTE 2

706E	23		INC	HL	
706F	3E20		LD	A,32	
7071	BE		CP	(HL)	
7072	20F6		JR	NZ,L2	
7074	3E00		LD	A,0	
7076	CDB4BB		CALL	OBBB4H	;CANAL 0
7079	212072		LD	HL,TEXT3	
707C	7E	L3:	LD	A,(HL)	
707D	CD5ABB		CALL	OBB5AH	;TEXTE 3
7080	23		INC	HL	
7081	3E20		LD	A,32	
7083	BE		CP	(HL)	
7084	20F6		JR	NZ,L3	
7086	CD06BB		CALL	OBB06H	;ATTENTE FRAPPE TOUCHE
7089	0601		LD	B,1	
708B	0E02		LD	C,2	
708D	CDB7BB		CALL	OBBB7H	;ECHANGE FEN1 <-> FEN2
7090	C34A70		JR	DEBUT	
7093	C9		RET		

END

--- LISTING DES CODES ---

!		!		!
!	ZONE DES CODES	!	NOM	!
!		!		!
!		!		!
!	7200 : 0A 09 46 45 4E 45 56 52 : ..FENETR	!	TEXT1	!
!	7208 : 45 09 31 20 00 00 00 00 : E.1.....	!		!
!		!		!
!	7210 : 0A 09 46 45 4E 45 54 52 : ..FENETR	!	TEXT2	!
!	7218 : 45 09 32 20 00 00 00 00 : E.2.....	!		!
!		!		!
!	7220 : 1F 0E 02 50 4F 55 52 09 : ...POUR.	!	TEXT3	!
!	7228 : 45 43 48 41 4E 47 45 52 : ECHANGER	!		!
!	7230 : 09 4C 45 53 09 46 45 4E : .LES.FEN	!		!
!	7238 : 45 54 52 45 53 1F 0E 04 : ETRES...	!		!
!	7240 : 41 50 50 55 59 45 5A 09 : APPUYEZ.	!		!
!	7248 : 53 55 52 09 55 4E 45 09 : SUR.UNE.	!		!
!	7250 : 54 4F 55 43 48 45 2E 2E : TOUCHE..	!		!
!	7258 : 2E 2E 00 00 00 00 00 00 :	!		!
!		!		!

=== EXEMPLE 2 ===

ORG 7000H

LOAD \$

POS: EQU 7100H

7000	CD4EBB	CALL	0BB4EH	;INITIALISATION
7003	CD06BB	CALL	0BB6CH	;CLS
7006	3E01	LD	A,1	
7008	CD9FBB	CALL	0BB9FH	;MODE TRANSPARENT
700B	CD0071	CALL	POS	
700E	3E4F	LD	A,"O"	
7010	CD5DBB	CALL	0BB5DH	;AFFICHAGE DE "O"
7013	CD0071	CALL	POS	
7016	3ECC	LD	A,204	
7018	CD5DBB	CALL	0BB5DH	;AFFICHAGE DE "/"
701B	C9	RET		
		LOAD	7100H	
701C	260A	LD	H,10	
701E	2E0A	LD	L,10	
7020	CD75BB	CALL	0BB75H	;POSITION CURSEUR
7023	C9	RET		
		END		

=== EXEMPLE 3 ===

Cet exemple consiste à redéfinir les caractères 250, 251, 252 et 253, puis à les afficher en formant un carré qui fera apparaître alors un personnage fort impressionnant....

Adresses concernées:

_ 7500H codes utilisés pour l'affichage.

_ 8000H zone de stockage des matrices venant de la ROM.

_ 7600H zone des valeurs des nouvelles matrices.

LISTING DES ZONES CODES :

7500H : 1F 13 0C FA FB 1F 13 0D

7508H : FC FD 20 00 00 00 00 00

7600H : 00 03 73 73 21 3F 3F 03

7608H : 00 C0 CE CE 84 FC FC C0

7610H : 03 03 03 0F 0F 0C 0C 3C

7618H : C0 C0 C0 F0 F0 30 30 3C

7620H : FF 00 00 00 00 00 00 00

ORG 7000H

LOAD \$

CODES: EQU 7500H

CARAC: EQU 7600H

MATRI: EQU 8000H

```
7000 CD4EBB      CALL 0BB4EH ;INITIALISATION
7003 CD6CBB      CALL 0BB6CH ;CLS ECRAN
7006 11FA00      LD DE,250
7009 210080      LD HL,MATRI
700C CDABBB      CALL 0BBABH ;STOCKAGE MATRICES EN RAM

700F 010080      LD BC,MATRI
7012 210076      LD HL,CARAC
7015 7E REDEF:   LD A,(HL) ;
7016 02          LD (BC),A ;REDEFINITION
7017 03          INC BC ;
7018 23          INC HL ; DES
7019 3EFF        LD A,255 ;
701B BE          CP (HL) ; MATRICES
```

701C 20F7		JR	NZ,REDEF ;
701E 210075		LD	HL,CODES
7021 7E	AFFIC:	LD	A,(HL)
7022 CD5ABB		CALL	0BB5AH ;AFFICHAGE NOUVEAUX CARACT.
7025 23		INC	HL
7026 3E20		LD	A,32
7028 BE		CP	(HL)
7029 20F6		JR	NZ,AFFIC
702B CD06BB		CALL	0BB06H ;ATTENTE FRAPPE TOUCHE
702E C9		RET	
END			

CHAPITRE 6 -

L'ECRAN GRAPHIQUE

I - GENERALITES:

L'écran graphique du CPC comporte 256000 points soit 640 dans la longueur sur 400 dans la hauteur. L'unité de base de l'écran ou pixel est formé par plusieurs points dont le nombre dépend du mode choisi:

- __ mode 0: 4 points dans a largeur, 2 dans la hauteur.
- __ mode 1: 2 points dans le largeur, 2 dans la hauteur.
- __ mode 2: 1 point dans la largeur, 2 dans la hauteur.

Quel que soit le mode, le système de repérage reste le même. Il comporte une origine située, à l'initialisation, en bas et à gauche de l'écran, dont les coordonnées sont 0,0; et deux axes: un axe horizontal gradué de 0 à 639 (abscisse: X) et un axe vertical gradué de 0 à 399 (ordonnée: Y)...

Deux systèmes de coordonnées peuvent être utilisés:

- __ coordonnées absolues: déplacement par rapport à l'origine.
- __ coordonnées relatives: déplacement relatif par rapport à la dernière position du curseur graphique.

L'origine peut être redéfinie par l'utilisateur à n'importe quel endroit de l'écran. Dans ce cas les coordonnées absolues seront exprimées par rapport à cette nouvelle origine...

L'utilisateur peut également créer une seule fenêtre graphique en indiquant ses limites internes. Il est à noter que lorsque des routines permettant l'affichage de traits, de caractères ou de points seront appelées, seules les points intérieurs à la fenêtre seront affichés

L'écran graphique permet également l'affichage de caractères. Leur coin en haut et à gauche se situera alors à la position du curseur.

I - ROUTINES UTILISEES:

Ces routines sont, comme dans le chapitre précédent, présentées par ordre de placement dans la RAM. Il sera précisé pour chacune d'elle, si elle figure dans l'exemple d'utilisation qui se trouve à la fin de ce chapitre ...

```

=====

! ROUTINE ! FONCTIONS: Initialisation de l'écran: suppression de !
!
!      ! la fenêtre, encr 0 pour le papier, encr 1 pour le !
! BBBA H ! stylo et placement en bas et à gauche du curseur.    !
! 48058 D !
!
! _____!
!
!              PARAMETRES              ! INDICATEURS! REG. !
!
!      ENTREE              SORTIE      ! AFFECTES. ! AFF. !
!
!-----!-----!-----!-----!
!
!              !              !   S=1   !   A   !
!
!              !              !   Z=0   !   BC  !
!
!              !              !   N=1   !   DE  !
!
!              !              !   C=1   !   HL  !
!
!              !              !          !      !
!
!              !              !          !      !
!
!-----!
! NOTES: Cette routine n'efface pas l'écran. Voir exemple.    !
!
!
=====

```

```

=====

! ROUTINE ! FONCTIONS: Positionnement du curseur à une position !
!
! absolue par rapport à l'origine normale ou celle qu'à !
! BBC0 H ! redéfinie l'utilisateur. !
! 48064 D !
! _____!
!
! PARAMETRES ! INDICATEURS! REG. !
!
! ENTREE SORTIE ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! DE -> coordonnée X ! ! ! DE !
! ! ! ! HL !
! HL -> coordonnée Y ! ! ! !
! ! ! !
! ! ! !
! ! ! !
!-----!
! NOTES: Le système ne corrige pas l'envoi d'une coordonnée !
! illégale. Voir exemple. !
=====

```

```

=====

! ROUTINE ! FONCTIONS: Déplacement du curseur vers une position !
!          ! relative à la position en cours.                      !
! BBC3 H !                                                         !
! 48067 D !                                                         !
! _____!

!          PARAMETRES          ! INDICATEURS! REG. !
!          ENTREE              SORTIE    ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! DE -> Déplac. X du curseur !          ! Z=1    ! DE !
!      ( valeur signée )    !          !      ! HL !
!                          !          ! reste à 0 !    !
! HL -> Déplac. Y du curseur !          !          !    !
!      ( valeur signée )    !          !          !    !
!                          !          !          !    !
!-----!

! NOTES: Le système ne corrige pas un déplacement illégal. Voir !
! exemple.                                                         !
=====

```

```

=====

! ROUTINE ! FONCTIONS: Renseigne sur la position du curseur par !
!          ! rapport à l'origine en cours.                      !
! BBC6  H !                                                    !
! 48070 D !                                                    !
! _____! _____!
!
!          PARAMETRES          !INDICATEURS! REG. !
!
!          ENTREE              SORTIE    ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
!
!          ! DE -> coord. X !          !          !
!          !                  !          !          !
!          ! HL -> coord. Y !          !          !
!          !                  !          !          !
!          !                  !          !          !
!          !                  !          !          !
!          !                  !          !          !
!-----!-----!-----!-----!
! NOTES:                                     !
!
!
=====

```

```

=====

! ROUTINE ! FONCTIONS: Positionnement de l'origine par rapport à !
!
! l'origine initiale(0,0 en bas et à gauche de l'écran) !
! BBC9 H !
! 48073 D !
! _____ !
!
! PARAMETRES ! INDICATEURS! REG. !
!
! ENTREE SORTIE ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! DE -> coord. X de la nouv. ! ! DE !
! origine. ! ! HL !
! ! !
! HL -> coord. Y de la nouv. ! !
! origine. ! !
! ! !
!-----!
! NOTES: Le système ne corrige pas l'envoi d'une coordonnée !
! illégale. !
=====

```

```

=====

! ROUTINE ! FONCTIONS: Renseigne sur la position de l'origine par !
!          ! rapport à l'origine initiale ( 0,0 en bas et à !
! BBCC H ! gauche de l'écran ).                                !
! 48076 D !                                                    !
! _____!_____!
!
!          PARAMETRES          !INDICATEURS! REG. !
!
!          ENTREE              SORTIE      ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
!
!          ! DE -> coord. X !          !
!
!          ! origine en cours!          !
!
!          !          !          !
!
!          ! HL -> coord. Y !          !
!
!          ! origine en cours!          !
!
!          !          !          !
!-----!
! NOTES:                                !
!
!
=====

```

```

=====

! ROUTINE ! FONCTIONS: Positionnement des limites verticales de la!
!
!      ! fenêtre par rapport à l'origine initiale.Si elles sont!
! BBCF H ! illégales le système les réduit ... Le bord gauche est!
! 48079 D ! indiqué par la plus petite des deux limites.      !
! _____!
!
!      PARAMETRES      ! INDICATEURS! REG. !
!
!      ENTREE      SORTIE      ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! DE -> bord 1 (a)      !      ! Z=1 ; C=1 ! A !
! HL -> bord 2 (b)      !      !      ! DE !
!      !      !      ! reste à 0 ! HL !
!
! MODE 0: a= 2*m ; b=2*n-1 !      !      !
!
! MODE 1: a= 4*m ; b=4*n-1 !      !      !
!
! MODE 2: a= 8*m ; b=8*n-1 !      !      !
!-----!
!
! NOTES: Il n'est possible de créer qu'une seule fenêtre. Voir !
! exemple.      !
=====

```



```

=====

! ROUTINE ! FONCTIONS: Renseigne sur les bords gauche et droit de !
!
! la fenêtre. Les valeurs sont données par rapport à !
! BBD5 H ! l'origine initiale ( 0,0 en bas et à gauche ). !
! 48085 D !
!
! _____!
!
! PARAMETRES !INDICATEURS! REG. !
!
! ENTREE SORTIE ! AFFECTES. ! AFF. !
!
!-----!-----!-----!-----!
!
! DE -> coord. X ! S=1 ! A !
!
! du bord gauche ! ! !
!
! ! ! reste à 0 ! !
!
! HL -> coord. X ! ! !
!
! du bord droit ! ! !
!
! ! ! !
!
!-----!
! NOTES: !
!
!
=====

```

=====

! ROUTINE ! FONCTIONS: Renseigne sur bords haut et bas de la !
! fenêtre. Les valeurs sont exprimées par rapport à !
! BBD8 H ! l'origine initiale (0,0 en bas et à gauche). !
! 48088 D !
! _____ !

! PARAMETRES ! INDICATEURS! REG. !

! ENTREE SORTIE ! AFFECTES. ! AFF. !

!-----!-----!-----!-----!

! DE -> bord haut ! Z=1 ! !

! de la fenêtre ! ! !

! ! reste à 0 ! !

! HL -> bord bas ! ! !

! de la fenêtre ! ! !

! ! ! !

!-----!

! NOTES: !

! !

=====

```

=====

! ROUTINE ! FONCTIONS: Effaçage de la fenêtre graphique. Celle-ci !
!
! se retrouve alors dans la couleur du papier. Si la !
! BBDB H ! fenêtre graphique n'est pas définie, l'écran entier !
! 48091 D ! est effacé. !
! _____ !
!
! PARAMETRES ! INDICATEURS! REG. !
!
! ENTREE SORTIE ! AFFECTES. ! AFF. !
!-----!-----!-----!
!
! ! ! Z=1 ; N=1 ! A !
!
! ! ! BC !
!
! ! ! reste à 0 ! DE !
!
! ! ! HL !
!
! ! !
!
! ! !
!-----!
! NOTES: Voir exemple. !
!
!
=====

```



```

=====

! ROUTINE ! FONCTIONS: Renseigne sur le numéro d'encre du stylo. !

!           !                                           !

! BBE1  H !                                           !

! 48097 D !                                           !

! _____!_____!

!           PARAMETRES           !INDICATEURS! REG. !

!           ENTREE           SORTIE   ! AFFECTES. ! AFF. !

!-----!-----!-----!-----!

!           ! A -> numéro de ! Z=1 ; N=1 !           !

!           ! l'encre du stylo!           !           !

!           !           ! reste à 0 !           !

!           !           !           !           !

!           !           !           !           !

!           !           !           !           !

!-----!

! NOTES:                                           !

!                                           !

=====

```

```

=====

! ROUTINE ! FONCTIONS: Choix du numéro d'encre du papier. En cas !
!
! de valeur illégale, même remarque que pour la routine !
! BBE4 H ! BBDE. !
! 48100 D !
! _____!
!
! PARAMETRES ! INDICATEURS! REG. !
!
! ENTREE SORTIE ! AFFECTES. ! AFF. !
!-----!-----!-----!
! A -> numéro de l'encre du ! S=1 ; 0=1 ! A !
! papier. ! !
! ! C=0 ! !
! ! !
! ! bit 3=1 ! !
! ! !
!-----!
! NOTES: Voir exemple. !
!
!
=====

```

```

=====

! ROUTINE ! FONCTIONS: Renseigne sur le numéro d'encre du papier. !

!           !                                     !

! BBE7  H !                                     !

! 48103 D !                                     !

! _____ !

!                                     PARAMETRES          !INDICATEURS! REG. !

!           ENTREE              SORTIE      ! AFFECTES. ! AFF. !

!-----!-----!-----!-----!

!                                     ! A -> numéro de ! Z=1 ; N=1 !      !

!                                     ! l'encre utilisée!          !      !

!                                     ! pour le papier ! reste à 0 !      !

!                                     !              !              !      !

!                                     !              !              !      !

!                                     !              !              !      !

!-----!

! NOTES:                                     !

!                                     !

=====

```


=====

! ROUTINE ! FONCTIONS: Place un pixel à une position absolue par !
! par rapport à l'origine en cours. Si les coordonnées !
! BBEA H ! sont en dehors de la fenêtre, la routine n'a aucun !
! 48106 D ! effet. !

! _____ !

! PARAMETRES ! INDICATEURS ! REG. !

! ENTREE SORTIE ! AFFECTES. ! AFF. !

!-----!-----!-----!-----!

! DE -> coord. X du pixel ! oui ! A !

! ! ! BC !

! HL -> coord. Y du pixel ! ! DE !

! ! ! HL !

! ! ! !

! ! ! !

!-----!

! NOTES: Voir exemple. !

! !

=====

```

=====

! ROUTINE ! FONCTIONS: Place un pixel à une position relative à !
!          ! la dernière position du curseur. Si les coordonnées !
! BBED H ! sont en dehors de l'écran, la routine n'a aucun effet !
! 48109 D !
! _____!
!
!          PARAMETRES          !INDICATEURS! REG. !
!
!          ENTREE          SORTIE    ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! DE -> déplacement X      !          ! oui    ! A    !
!      ( valeur signée )   !          !          ! BC   !
!                          !          !          ! DE   !
! HL -> déplacement Y      !          !          ! HL   !
!      ( valeur signée )   !          !          !      !
!                          !          !          !      !
!-----!
! NOTES: Voir exemple.
!
=====

```

```

=====

! ROUTINE ! FONCTIONS: Indication du numéro de l'encre d'un pixel !

!          ! situé à une position absolue par rapport à l'origine !

! BBF0 H ! initiale.                                     !

! 48112 D !                                              !

! _____!_____!

!          PARAMETRES          !INDICATEURS! REG. !

!          ENTREE              SORTIE    ! AFFECTES. ! AFF. !

!-----!-----!-----!-----!

! DE -> coord. X du pixel    ! A -> numéro de ! Z=1 ; N=1 ! BC !

!                          ! l'encre du !      ! DE !

! HL -> coord. Y du pixel    ! pixel    ! reste à 0 ! HL !

!                          !          !          !          !

!                          !          !          !          !

!                          !          !          !          !

!-----!

! NOTES: En cas de valeur illégale des coordonnées le registre A !

! contient en sortie le numéro de l'encre du papier.             !

=====

```

```

=====

! ROUTINE ! FONCTIONS: Indication du numéro de l'encre d'un pixel !

!          ! situé à une position relative à la dernière position !

! BBF3 H ! du curseur. !

! 48115 D ! !

! _____ !

!          PARAMETRES          ! INDICATEURS ! REG. !

!          ENTREE          SORTIE ! AFFECTES. ! AFF. !

!-----!-----!-----!-----!

! DE -> déplac. X du curseur ! A -> numéro de ! Z=1 ; N=1 ! BC !

!      ( valeur signée )      !      l'encre du !      DE !

!          !      pixel      ! reste à 0 ! HL !

! HL -> déplac. Y du curseur !          !          !

!      ( valeur signée )      !          !          !

!          !          !          !

!-----!

! NOTES: En cas de valeur illégale du déplacement, A contient le !

! numéro de l'encre du papier. !

=====

```

```

=====

! ROUTINE ! FONCTIONS: Traçage d'une ligne depuis la dernière !
!
! position du curseur jusqu'à une position absolue par !
! BBF6 H ! rapport à l'origine en cours. !
! 48118 D !
! _____ !
!
! PARAMETRES ! INDICATEURS! REG. !
!
! ENTREE SORTIE ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! DE -> coord. X du dernier ! ! Z=1 ; 0=1 ! A !
! point de la ligne ! ! ! BC !
! ! ! reste à 0 ! DE !
! HL -> coord. Y du dernier ! ! ! HL !
! point de la ligne ! ! ! !
! ! ! !
!-----!
! NOTES: Seuls les points internes à la fenetre sont tracés. Voir !
! exemple. !
=====

```

```

=====

! ROUTINE ! FONCTIONS: Traçage d'une ligne depuis la dernière !
!          ! position du curseur jusqu'à une position relative à !
! BBF9 H ! cette dernière position.                               !
! 48121 D !                                                         !
! _____!_____!
!
!          PARAMETRES          ! INDICATEURS! REG. !
!
!          ENTREE          SORTIE    ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! DE -> coord. relative X du!          ! Z=1 ; 0=1 ! A !
!          dernier point de la!          !          ! BC !
!          ligne (valeur signée)!          ! reste à 0 ! DE !
! HL -> coord. relative Y du!          !          ! HL !
!          dernier point de la!          !          !    !
!          ligne (valeur signée)!          !          !    !
!-----!
! NOTES: Seuls les points internes à la fenêtre sont tracés. Voir !
! exemple.                                                         !
=====

```

```

=====

! ROUTINE ! FONCTIONS: Affichage d'un caractère. Le coin gauche !
!          ! et haut du caractère correspond à la position du !
! BBFC H ! curseur qui se déplace après l'affichage de 32, 16 ou !
! 48124 D ! 8 points selon le mode en cours: 0, 1 ou 2.          !
! _____! _____!
!
!          PARAMETRES          ! INDICATEURS! REG. !
!
!          ENTREE          SORTIE          ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
!
!          ! A -> code ASCII ! Z=1 ! A !
!
!          ! du caractère ! ! BC !
!
!          ! ! reste à 0 ! DE !
!
!          ! ! ! HL !
!
!          ! ! ! !
!
!          ! ! ! !
!-----!
! NOTES: Les codes de controle ne sont pas obéis mais affichés. !
! Voir exemple.          !
=====

```

=== EXEMPLE D'UTILISATION ===

L'exemple suivant se propose d'utiliser un maximum de routines étudiées. Pour cela, une fenêtre graphique sera définie puis coloriée. Après détermination de la couleur du stylo, un triangle sera tracé à l'intérieur de la fenêtre. Après avoir redéfini les couleurs des encres du papier et du stylo, le mot " TRIANGLE " sera affiché à l'écran en décalant de 8 points vers le bas chacun de ses caractères...

```

                                ORG    7000H

                                LOAD   $

                                STO:   EQU    7400H

                                TEXT:   EQU    74FFH

7000  CD6CBB                CALL  OBB6CH    ;CLS ECRAN

7003  CDBABB                CALL  OBBBAH    ;INITIALISATION

7006  21FF74                LD     HL,TEXT

7009  220074                LD     (STO),HL

700C  117400                LD     DE,116

700F  210802                LD     HL,523

7012  CDCFBB                CALL  OBBCFH    ;LIMITES VERTIC. FENETRE

7015  114800                LD     DE,72

7018  214801                LD     HL,328

701B  CDD2BB                CALL  OBBD2H    ;LIMITES HORIZ. FENETRE

```


701E	3E02	LD	A,2	
7020	CDE4BB	CALL	0BBE4H	;NUMERO ENCRE PAPIER
7023	CDD8BB	CALL	0BBD8H	;CLS FENETRE
7026	3E03	LD	A,3	
7028	CDDEBB	CALL	0BBDEH	;NUMERO ENCRE STYLO
702B	119C00	LD	DE,156	
702E	217000	LD	HL,112	
7031	CDC0BB	CALL	0BBC0H	;POSIT. ABSOLUE CURSEUR
7034	113C00	LD	DE,60	
7037	219A00	LD	HL,154	
703A	CDF9BB	CALL	0BBF9H	;TRACAG. LIGNE REL/CURS.
703D	11E801	LD	DE,488	
7040	217000	LD	HL,112	
7043	CDF6BB	CALL	0BBF6H	;TRACAG. LIGNE POS. ABS.
7046	119C00	LD	DE,156	
7049	217000	LD	HL,112	
704C	CDF6BB	CALL	0BBF6H	;TRACAG. LIGNE POS. ABS.
704F	3E01	LD	A,1	
7051	CDE4BB	CALL	0BBE4H	;NUMERO ENCRE PAPIER TEXTE
7054	11AC00	LD	DE,172	

7057	219000		LD	HL,144	
705A	CDC3BB		CALL	0BBC3H	;DEPL. RELATIF CURSEUR
705D	3E03		LD	A,3	
705F	CDDEBB		CALL	0BBDEH	;NUMERO ENCRE STYLO TEXTE
7062	2A0074	ECRIT:	LD	HL,(STO)	
7065	23		INC	HL	
7066	220074		LD	(STO),HL	
7069	7E		LD	A,(HL)	
706A	0620		LD	B,32	
706C	B8		CP	B	
706D	CA7E70		JP	Z,0707EH	;TEST FIN DE TEXTE
7070	CDFCBB		CALL	0BBFCH	;AFFICHAGE TEXTE
7073	110000		LD	DE,0	
7076	21F8FF		LD	HL,0-8	
7079	CDC3BB		CALL	0BBC3H	;DEPLAC. RELAT. CURSEUR
707C	18E4		JR	ECRIT	
707E	CD06BB		CALL	0BB06H	;ATTENTE FRAPPE TOUCHE
7081	C9		RET		

END

LISTING CODES:

7500: 54 52 49 41 4E 47 4C 45 : TRIANGLE

7508: 20 00 00 00 00 00 00 00 :

CHAPITRE 7 -

GESTION DE LA MEMOIRE ECRAN

I - GENERALITES :

a) Organisation de la mémoire écran:

La zone mémoire de l'écran du CPC occupe une plage de 16K de la RAM située à l'initialisation à partir de C000H jusqu'à FFFFH; elle peut être déplacée par l'utilisateur, s'il le désire, vers une autre adresse de la RAM. En fait seule la partie allant de 4000H à 7FFFH convient car, située ailleurs, la mémoire écran effacerait des zones de données utilisées par le système ... Il a été dit dans les chapitres précédents que l'écran de l'Amstrad est formé de 640 points pour la longueur, sur 400 pour la hauteur.

Quel que soit le mode choisi l'écran comporte 200 lignes de pixels de 2 points de haut.

Sachant que le mémoire occupe 16K nous pouvons en déduire que chacune des lignes sera représentée par 80 octets ($200 \times 80 = 16000$). En partageant les 16K de mémoire en 8 groupes de 2K, chacun d'entre eux peut alors contenir les octets nécessaires à la représentation de 25 lignes de pixels ($25 \times 80 = 2000$).

Or l'écran est justement formé de 25 lignes de caractères de 8 pixels de haut, ce qui explique que le premier groupe de 2K contient les 80 octets représentant les premières lignes de pixels de chacune des 25 lignes de caractères.

De même, le deuxième groupe de 2K contient les 80 octets représentant les deuxièmes lignes de pixels des 25 lignes de caractères et ainsi de suite jusqu'au huitième groupe de 2K qui représente les huitièmes et dernières lignes de pixels des 25 lignes de caractères. Les adresses de début et de fin des 8 groupes sont: de C000H à C7CFH, de C800H à CFCFH, de D000H à D7CFH, de D800H à DFCFH, de E000H à EFCFH, de E800H à E8CFH, de F000H à FFCFH et de F800H à FFCFH. Les 48 octets situés à la fin de chacun des 8 groupes ne sont pas utilisés.

=== CARTE MEMOIRE ECRAN ===

<----- 80 octets ----->		
caract		de C000H à C0AFH
ligne 1		de C800H à C8AFH
		de D000H à D0AFH
		de D800H à D8AFH
		de E000H à E0AFH
		de E800H à E8AFH
		de F000H à F0AFH
		de F800H à F8AFH
caract		de C050H à C09FH
ligne 2		de C850H à C89FH
.		
.		
.		
.		
caract		
ligne 24		
		de FF30H à FF7FH
caract		de C780H à C7CFH
ligne 25		de CF80H à CFCFH
		de D780H à D7CFH
		de DF80H à DFCFH
		de E780h à E7CFH
		de EF80H à EFCFH
		de F780H à F7CFH
		de FF80H à FFCFH

Ce tableau permet de comprendre comment se font les conversions de coordonnées de caractères ou de pixels, en adresses de la mémoire écran. Mais avant de donner les formules permettant ces calculs, il est nécessaire de parler de la possibilité qu'offre le CPC de décaler l'affichage à l'écran. En effet, si à l'initialisation, le premier octet à être affiché est celui dont l'adresse est C000H, il est possible de faire débiter l'affichage par n'importe lequel des octets du premier groupe de 2K; ainsi, le premier caractère affiché sera celui dont le coin haut et gauche se trouve à cette adresse.

Le décalage de l'écran est donc une valeur modulo 800H. Pour effectuer un scrolling de l'écran d'une ligne de caractère vers le haut ou vers le bas, il suffit alors de décaler celui-ci de 80 ou de -80. Si DECA représente le décalage de l'écran (0 à l'initialisation), NO le nombre d'octets représentant un caractère et NP le nombre de pixels par octet selon le mode choisi (mode 0, NO=4, NP=2; mode 1, NO=2, NP=4; mode 2, NO=1, NP=8) alors les conversions de coordonnées en adresses de la mémoire écran s'effectueront de la manière suivante:

- Soit un caractère de coordonnées (C,L) par rapport au coin haut et à gauche de l'écran (0,0).

$$\text{ADRESSE MEMOIRE} = \text{C000H} + (\text{L} * 80\text{H}) + (\text{C} * \text{NO}) + \text{DECA}$$

- Soit un pixel de coordonnées (X,Y) par rapport au coin bas et à gauche de l'écran (0,0).

$$\text{ADRESSE MEMOIRE} = (((199 - Y) \setminus 8) * 80 + (X \setminus \text{NP}) + \text{DECA}) \bmod 800\text{H} \\ + ((199 - Y) \bmod 8) * 800\text{H} + \text{C000H}$$

b) Mode:

Le choix du mode, parmi les 3 proposés par l'Amstrad est très important, car il conditionne plusieurs éléments de l'affichage comme la taille des caractères, le nombre de couleurs utilisables, la taille des pixels, le nombre de caractères affichés par ligne, les adresses mémoire des pixels ainsi que le codage des couleurs qui sera exposé dans le paragraphe suivant. Voici les principales caractéristiques de chaque mode résumées dans un tableau:

	MODE 0	MODE 1	MODE 2
CARACTERES AFFICHES	20 sur 25	40 sur 25	80 sur 25
PIXELS AFFICHES	160 sur 200	320 sur 200	640 sur 200
TAILLE PIXELS (en points)	4 sur 2	2 sur 2	1 sur 2
NOMBRE OCTETS par caractère	4	2	1
NOMBRE PIXELS par octet	2	4	8
COULEURS UTILISABLES	16	4	2

Il résulte de ce tableau que, quel que soit le mode en cours, un caractère est toujours formé de 8 pixels sur 8.

c) Codage des couleurs:

Le codage des couleurs est une tâche particulièrement délicate car il diffère suivant le mode choisi. Afin de faciliter le travail de l'utilisateur, celui-ci trouvera, pour chaque mode, une première partie expliquant le procédé utilisé pour le codage, puis un tableau permettant d'attribuer aux pixels de chaque octet, toutes les couleurs utilisables. Le principe général consiste à donner aux bits représentant chaque pixel, le numéro d'encre que l'utilisateur désire lui voir porter.

* MODE 0:

Dans ce mode chaque octet de la mémoire écran affichera 2 pixels de 4 points. Un pixel sera donc codé sur 4 bits, permettant ainsi d'obtenir des valeurs allant de 0 à 15. Ce qui explique que dans ce mode, 16 couleurs peuvent être attribuées à chaque pixel. Si les bits de l'octet sont repérés par leur numéro d'ordre, soit de 0 à 7 de droite vers la gauche, le codage du pixel gauche se fera avec les bits 1, 5, 3 et 7 tandis que celui du pixel droit se fera avec les bits 0, 4, 2 et 6:

Pixel: GAUCHE DROIT

Bit No:	1	5	3	7	0	4	2	6
---------	---	---	---	---	---	---	---	---

Valeur:	8	4	2	1	8	4	2	1
---------	---	---	---	---	---	---	---	---

Supposons que nous voulons colorer le pixel de gauche avec l'encre numéro 11 et celui de droite avec l'encre numéro 5. Il faudra mettre les bits 1, 3, 7 puis les bits 4 et 6 à l'état 1. Pour calculer alors la valeur à donner à l'octet représentant les 2 pixels il faut replacer tous ces bits dans leur ordre originel avec l'état trouvé plus haut:

Valeur:	8	4	2	1	8	4	2	1
---------	---	---	---	---	---	---	---	---

Bit No:	7	6	5	4	3	2	1	0
---------	---	---	---	---	---	---	---	---

Etat:	1	1	0	1	1	0	1	0
-------	---	---	---	---	---	---	---	---

$$8 + 4 + 1 = \text{DH}$$

$$8 + 2 = \text{AH}$$

La valeur à placer dans l'octet est donc DAH. Le tableau suivant donne la valeur de l'octet pour toutes les combinaisons d'encres...

CODAGE DES ENCRE : MODE 0

		PIXEL DROIT															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P I X E L	0	00	40	04	44	10	50	14	54	01	41	05	45	11	51	15	21
	1	80	C0	84	C4	90	D0	94	D4	81	C1	85	C5	91	D1	95	A1
	2	08	48	0C	4C	18	58	1C	5C	09	49	0D	4D	19	59	1D	5D
	3	88	C8	8C	CC	98	D8	9C	DC	89	C9	8D	CD	99	D9	9D	DD
	4	20	60	24	64	30	70	34	74	21	61	25	65	31	71	35	75
	5	A0	E0	A4	E4	B0	F0	B4	F4	A1	E1	A5	E5	B1	F1	B5	F5
	6	28	68	2C	6C	38	78	3C	7C	29	69	2D	6D	39	79	3D	7D
	7	A8	E8	AC	EC	B8	F8	BC	FC	A9	E9	AD	ED	B9	F9	BD	FD
	8	02	42	06	46	12	52	16	56	03	43	07	47	13	53	17	57
	9	82	C2	86	C6	92	D2	96	D6	83	C3	87	C7	93	D3	97	D7
	10	0A	4A	0E	4E	1A	5A	1E	5E	0B	4B	0F	4F	1B	5B	1F	5F
	11	8A	CA	8E	CE	9A	DA	9E	DE	8B	CB	8F	CF	9B	DB	9F	DF
	12	22	62	26	66	32	72	36	76	23	63	27	67	33	73	37	77
	13	A2	E2	A6	E6	B2	F2	B6	F6	A3	E3	A7	E7	B3	F3	B7	F7
	14	2A	6A	2E	6E	3A	7A	3E	7E	2B	6B	2F	6F	3B	7B	3F	7F
	15	AA	EA	AE	EE	BA	FA	BE	FE	AB	EB	AF	EF	BB	FB	BF	FF

Pour trouver la valeur correspondant à l'exemple précédent soit: encre 11 pour le pixel gauche et encre 5 pour le pixel droite, il suffit de se placer sur la ligne 11, puis sur la colonne 5. Nous pouvons alors lire dans cette case la valeur DAH....

* MODE 1:

Dans ce mode, un octet de la mémoire écran permet de placer les couleurs des encres de 4 pixels de 2 points chacun. Les pixels seront donc codés sur 2 bits, permettant ainsi de placer des valeurs allant de 0 à 3. Voilà pourquoi, en mode 1, 4 couleurs sont disponibles en même temps à l'écran pour, chaque pixel.

Si les 8 bits de l'octet sont numérotés comme dans le cas du mode 0 et que de plus, les 4 pixels portent, de la gauche vers la droite, les lettres A, B, C et D, alors le pixel A sera représenté par les bits 3 et 7, B par les bits 2 et 6, C par les bits 1 et 5 et enfin D par les bits 0 et 4 comme le montre le schéma suivant:

Pixel:	A		B		C		D	
Bit No:	3	7	2	6	1	5	0	4
Valeur:	2	1	2	1	2	1	2	1

Si nous voulons attribuer les encres 2, 1, 0 et 3 respectivement aux pixels A, B, C et D, alors il faudra mettre à l'état 1 les bits 3, 6, 0 et 4. La valeur prise par l'octet représentant ces pixels à l'écran sera alors:

Valeur:	8	4	2	1	8	4	2	1
Bit No:	7	6	5	4	3	2	1	0
Etat:	0	1	0	1	1	0	0	1
	4 + 1 = 5H				8 + 1 = 9H			

La valeur à placer dans l'octet sera donc 59H. Le tableau suivant donne la valeur de l'octet pour toutes les combinaisons possibles.

CODAGES DES ENCREES : MODE 1

			PIXEL C=0				PIXEL C=1				PIXEL C=2				PIXEL C=3			
			PIXEL D				PIXEL D				PIXEL D				PIXEL D			
			0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
P	P	0	00	10	01	11	20	30	21	31	02	12	03	13	22	32	23	33
I	I																	
X	X	1	40	50	41	51	60	70	61	71	42	52	43	53	62	72	63	73
E	E																	
A	L	2	04	14	05	15	24	34	25	35	06	16	07	17	26	36	27	37
=																		
0	B	3	44	54	45	55	64	74	65	75	46	56	47	57	66	76	67	77
P	P	0	80	90	81	91	A0	B0	A1	B1	82	92	83	93	A2	B2	A3	B3
I	I																	
X	X	1	C0	D0	C1	D1	E0	F0	E1	F1	C2	D2	C3	D3	E2	F2	E3	F3
E	E																	
A	L	2	84	94	85	95	A4	B4	A5	B5	86	96	87	97	A6	B6	A7	B7
=																		
1	B	3	C4	D4	C5	D5	E4	F4	E5	F5	C6	D6	C7	D7	E6	F6	E7	D7
P	P	0	08	18	09	19	28	38	29	39	0A	1A	0B	1B	2A	3A	2B	3B
I	I																	
X	X	1	48	58	49	59	68	78	69	79	4A	5A	4B	5B	6A	7A	6B	7B
E	E																	
A	L	2	0C	1C	0D	1D	2C	3C	2D	3D	0E	1E	0F	1F	2E	3E	2F	3F
=																		
2	B	3	4C	5C	4D	5D	6C	7C	6D	7D	4E	5E	4F	5F	6E	7E	6F	7F
P	P	0	88	98	89	99	A8	B8	A9	B9	8A	9A	8B	9B	AA	BA	AB	BB
I	I																	
X	X	1	C8	D8	C9	D9	E8	F8	E9	F9	CA	DA	CB	DB	EA	FA	EB	FB
E	E																	
A	L	2	8C	9C	8D	9D	AC	BC	AD	BD	8E	9E	8F	9F	AE	BE	AF	BF
=																		
3	B	3	CC	DC	CD	DD	EC	FC	ED	FD	CE	DE	CF	DF	EE	FE	EF	FF

Pour retrouver la valeur de l'exemple précédent, il faut se placer dans le groupe de 4 lignes où $A=2$, puis sur la ligne où $B=1$, puis dans le groupe de 4 colonnes où $C=0$ et enfin lire la valeur dans la colonne où $D=3$ soit 59H...

* MODE 2 :

C'est le mode pour lequel le codage est le plus simple. En effet, dans ce cas, chaque octet de la mémoire écran représente 8 pixels d'un point chacun, dont l'encre sera fixée par l'état des 8 bits de l'octet. Il y a donc 2 couleurs disponibles qui portent les numéros 0 et 1. Ainsi, si les pixels sont numérotés de 1 à 8 de la gauche vers la droite et que les pixels 1, 3, 4, 6 et 8 doivent être mis dans l'encre numéro 1, alors la valeur de l'octet sera B5H car sa conversion en binaire est 1 0 1 1 0 1 0 1...

Quel que soit le mode en cours, la routine chargée de donner leur valeur aux octets de la mémoire écran nécessite l'envoi, dans le registre C d'un nombre que nous appellerons "cache". Ce cache permet de ne colorer que certains points des pixels dont le numéro d'encre est fourni par le codage de l'octet. Mais voyons plutôt un exemple:

OCTET				CACHE			
				Valeur:	8	4	2 1
Bit No:	1 5 3 7	0 4 2 6		Bit No:	7 6 5 4	3 2 1 0	
Point:	a b c d	e f g h		Etat:	0 1 0 1	1 0 1 0	
				4+1 = 5H 8+2 = AH			

Le mode en cours est le mode 0. Nous pouvons alors colorier les pixels de gauche et de droite (4 points chacun) avec, par exemple, les encres 5 et 9, ce qui donne, après lecture du tableau adéquat, la valeur E1H pour l'octet de la mémoire écran. Bien sur, les 4 points des 2 pixels porteront la même couleur.

Si nous désirons maintenant que seuls les points a, c, f et g soient de la couleur de l'encre précisée et les autres points, de la couleur du fond, il suffit de mettre à l'état 1 les bits correspondant à ces points.

D'après le graphique, ces bits portent les numéros 1, 3, 4 et 6. En les mettant à l'état 1 dans l'octet du cache, on obtient pour celui-ci, la valeur 5AH. Ce procédé permet en utilisant les modes 0 ou 1, d'obtenir, quand c'est possible, une meilleure résolution que celle proposée par le mode en cours.

d) Mode d'écriture graphique (MEG):

C'est la dernière particularité graphique de l'Amstrad. Elle permet de préciser la manière dont s'effectuera la mise en couleurs des pixels à l'écran. Cette manière dépendra d'une opération logique réalisée entre le numéro de l'encre de la position de l'écran où doit se placer le pixel et le numéro d'encre de ce pixel. Le numéro définitif de l'encre du pixel à afficher sera alors le résultat de l'opération logique.

Pour plus de commodités, nous appellerons:

AE, l'encre du fond ou ancienne encre.

NE, l'encre du pixel ou nouvelle encre.

ED, l'encre définitive du pixel.

Il existe 4 modes d'écriture graphique:

!	!	!	!
!	MEG	!	OPERATION
!	!	!	EFFET
!	!	!	!
!	!	!	!
!	0	!	aucune
!	!	!	ED = NE
!	!	!	!
!	1	!	OU EXCLUSIF
!	!	!	ED = AE XOR NE
!	!	!	!
!	2	!	ET
!	!	!	ED = AE AND NE
!	!	!	!
!	3	!	OU
!	!	!	ED = AE OU NE
!	!	!	!

Le MEG 0 est celui fixé à l'initialisation de la mémoire écran et pour lequel l'encre définitive est la nouvelle encre donnée au pixel. Mais pour mieux comprendre les effets obtenus, voici pour le mode écran 1, 3 tableaux indiquant le numéro de l'encre définitive que portera un pixel en fonction des anciennes et nouvelles encres et ce pour les MEG 1, 2 et 3 (NE en abscisse, AE en ordonnée):

MEG 1					MEG 2					MEG 3				
AE ↙ NE →	0	1	2	3	AE ↙ NE →	0	1	2	3	AE ↙ NE →	0	1	2	3
	0	1	2	3		0	1	2	3		0	1	2	3
0	0	1	2	3	0	0	0	0	0	0	0	1	2	3
1	1	0	3	2	1	0	1	0	1	1	1	1	2	3
2	2	3	0	1	2	0	0	2	2	2	2	1	2	3
3	3	2	1	0	3	0	1	2	3	3	3	1	2	3
OU EXCLUSIF					ET					OU				

Il s'agit en fait de tables de vérité, effectuées sur les 4 numéros pour les 3 opérations logiques indiquées. Ainsi, dans le MEG 1, un pixel de nouvelle encre 2, placé à un endroit de l'écran où l'encre est 1, sera affiché en définitive avec l'encre 3. Le principe est le même pour le mode écran 0, avec des tables de vérité de 16x16.

Il est à noter cependant que le MEG ne concerne que certaines routines graphiques qui seront précisées, mais en aucun cas l'affichage de texte...

II - ROUTINES UTILISEES:

Ces routines prennent place dans la zone de la RAM allant de BBFFH à BC62H.

```

=====

! ROUTINE ! FONCTIONS: Initialisation de la mémoire: mode 1, base !

!          ! mémoire à C000H, MEG 0, décalage à 0, CLS écran avec !

! BBFF H ! encre 0, mise à défaut des encres et de la période de !

! 48127 D ! clignotement.                                     !

! _____! _____!

!          PARAMETRES          ! INDICATEURS! REG. !

!          ENTREE              SORTIE      ! AFFECTES. ! AFF. !

!-----!-----!-----!-----!

!          !          ! Z=1 ; O=1 ! A  !

!          !          !          ! BC !

!          !          ! reste à 0 ! DE !

!          !          !          ! HL !

!          !          !          !    !

!          !          !          !    !

!-----!

! NOTES: Voir exemples 1 et 2.                                     !

!          !

=====

```



```

=====

! ROUTINE ! FONCTIONS: Fixe la valeur du décalage de l'écran. Ce !
!
! doit etre un multiple de 2 et en cas de valeur trop !
! BC05 H ! grande, le système utilise son modulo 2046 afin que !
! 48133 D ! l'écran démarre sur la 1ère ligne de pixel d'1 carac. !
! _____ !
!
! PARAMETRES ! INDICATEURS! REG. !
!
! ENTREE SORTIE ! AFFECTES. ! AFF. !
! .....! .....! .....! .....!
! HL -> décalage écran ! oui ! A !
! ! ! DE !
! ! ! HL !
! ! ! !
! ! ! !
! ! ! !
! ! ! !
! .....!
! NOTES: !
! !
=====

```


=====

! ROUTINE ! FONCTIONS: Fixe l'adresse de début de la zone mémoire !

! ! écran. !

! BC08 H !

! 48136 D !

! _____ !

! PARAMETRES ! INDICATEURS! REG. !

! ENTREE SORTIE ! AFFECTES. ! AFF. !

! -----! -----! -----! -----!

! A -> octet fort de ! oui ! A !

! l'adresse de la ! ! HL !

! mémoire écran. ! ! !

! ! ! !

! ! ! !

! ! ! !

! -----!

! NOTES: !

! !

=====

```

=====

! ROUTINE ! FONCTIONS: Indique le début de la zone mémoire écran. !

!           !                                     !

! BCOB  H !                                     !

! 48139 D !                                     !

! _____!_____!

!           PARAMETRES           !INDICATEURS! REG. !

!           ENTREE           SORTIE   ! AFFECTES. ! AFF. !

!-----!-----!-----!-----!

!           ! A -> octet fort !   non   ! HL !

!           !   le l'adresse!           !   !

!           !   de   la !           !   !

!           !   mémoire écran!           !   !

!           !           !           !   !

!           !           !           !   !

!-----!

! NOTES:                                     !

!                                           !

=====

```



```

=====

! ROUTINE ! FONCTIONS: Indique le mode écran en cours.      !

!           !                                                    !

! BC11  H !                                                    !

! 48145 D !                                                    !

! _____!_____!

!                                     PARAMETRES                !INDICATEURS! REG. !

!           ENTREE                SORTIE        ! AFFECTES. ! AFF. !

!-----!-----!-----!-----!

!                                     ! mode 0: A = 0 ->! Z=0 ; C=1 !      !

!                                     !           !           !           !

!                                     ! mode 1: A = 1 ->! Z=1 ; C=0 !      !

!                                     !           !           !           !

!                                     ! mode 2: A = 2 ->! Z=0 ; C=0 !      !

!                                     !           !           !           !

!-----!-----!-----!-----!

! NOTES:                                                    !

!                                                    !

=====

```



```

=====

! ROUTINE ! FONCTIONS: Renseigne sur les dernières lignes et !
!          ! colonnes utilisables à l'écran qui seront indiquées !
! BC17 H ! par les registres B et C, A contenant le mode écran. !
! 48151 D ! (coordonnées / coin haut à gauche repéré par 0,0) !
! _____!
!
!          PARAMETRES          ! INDICATEURS! REG. !
!
!          ENTREE              SORTIE      ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
!
!          ! mode 0: A=0  B=19  C=24 ->! Z=0 ; C=1 !      !
!
!          !
!          !
!
!          ! mode 1: A=1  B=39  C=24 ->! Z=1 ; C=0 !      !
!
!          !
!          !
!
!          ! mode 2: A=2  B=79  C=24 ->! Z=0 ; C=0 !      !
!
!          !
!          !
!-----!
!
! NOTES:
!
!
=====

```

```

=====

! ROUTINE ! FONCTIONS: Transforme une coordonnée de caractère en !
!
!      ! adresse mémoire écran. Les coordonnées sont données !
! BC1A H ! par rapport au coin haut à gauche repéré par 0,0.      !
! 48154 D !
! _____!
!
!              PARAMETRES              !INDICATEURS! REG. !
!
!      ENTREE              SORTIE      ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! H -> colonne du caractère ! HL -> adresse ! S=1 ! A !
!
!              !              !              !
! L -> ligne du caractère ! B -> nbre octets! reste à 0 !
!
!              ! par caractère!              !
!
!              !              !              !
!              !              !              !
!-----!
! NOTES: Voir formule de calcul dans le paragraphe précédent. !
!
!
=====

```

```

=====
! ROUTINE ! FONCTIONS: Transforme les coordonnées d'un pixel en !
!         ! adresse mémoire écran. Les coordonnées sont données !
! BC1D H ! par rapport au coin bas à gauche repéré par 0,0.      !
! 48157 D !                                                         !
! _____!_____!
!
!               PARAMETRES                !INDICATEURS! REG. !
!
!      ENTREE                SORTIE      ! AFFECTES. ! AFF. !
! .....! .....! .....! .....!
!
! DE -> coord. X du pixel      ! HL -> adresse      !   oui   !   A   !
!
!                               !   mémoire   !       !   DE   !
!
! HL -> coord. Y du pixel      ! C -> cache pixel!       !       !
!
!                               ! B -> nbre pixels!       !       !
!
!                               !   par octet -1!       !       !
!
!                               !               !       !       !
! .....! .....! .....! .....!
!
! NOTES: Voir formule de calcul dans le paragraphe précédent.      !
!
!      Voir exemple 2.                                              !
=====

```



```

=====

! ROUTINE ! FONCTIONS: Indique l'adresse de l'octet se trouvant à !
!
! gauche de l'octet dont l'adresse est indiquée dans le !
! BC23 H ! registre HL. !
! 48163 D ! !
! _____ !
!
! PARAMETRES ! INDICATEURS! REG. !
!
! ENTREE SORTIE ! AFFECTES. ! AFF. !
! .....! .....! .....! .....!
! HL -> adresse d'un octet ! HL -> adresse de! oui ! A !
!
! l'octet à ! !
!
! gauche ! !
!
! ! !
!
! ! !
!
! ! !
! .....!
! NOTES: !
!
!
=====

```

```

=====
! ROUTINE ! FONCTIONS: Indique l'adresse d'un octet se trouvant !
!
!      ! juste en dessous de l'octet dont l'adresse se trouve !
! BC26 H ! dans le registre HL.                                !
! 48166 D !                                                    !
! _____!
!
!          PARAMETRES          !INDICATEURS! REG. !
!
!      ENTREE          SORTIE    ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! HL -> adresse d'un octet  ! HL -> adresse de!   oui   ! A   !
!
!          !      l'octet en!          !      !
!
!          !      dessous  !          !      !
!
!          !          !          !      !
!
!          !          !          !      !
!
!          !          !          !      !
!-----!
! NOTES: Voir exemple 1.                                !
!
!
=====

```

```

=====

! ROUTINE ! FONCTIONS: Indique l'adresse de l'octet se trouvant !
!
!      ! juste au dessus de l'octet dont l'adresse est fournie !
!
! BC29 H ! au registre HL. !
!
! 48169 D !
!
! _____!
!
!          PARAMETRES          ! INDICATEURS! REG. !
!
!      ENTREE          SORTIE    ! AFFECTES. ! AFF. !
!
!-----!-----!-----!-----!
!
! HL -> adresse d'un octet    ! HL -> adresse de!    oui    ! A    !
!
!          !      l'octet au!          !      !
!
!          !      dessus    !          !      !
!
!          !          !          !      !
!
!          !          !          !      !
!
!          !          !          !      !
!
!-----!
!
! NOTES: !
!
!          !
=====

```

=====

! ROUTINE ! FONCTIONS: Indique le codage d'un octet afin que tous !

! ! les pixels de celui-ci soient dans la couleur voulue. !

! BC2C H ! !

! 48172 D ! !

! _____ !

! PARAMETRES ! INDICATEURS! REG. !

! ENTREE SORTIE ! AFFECTES. ! AFF. !

! ! ! !

! A -> numéro de l'encre ! A -> codage de ! oui ! !

! ! l'octet ! ! !

! ! ! ! !

! ! ! ! !

! ! ! ! !

! ! ! ! !

! !

! NOTES: !

! !

=====

```

=====

! ROUTINE ! FONCTIONS: Fournit le numéro d'encre du pixel de gauche
!
!          ! d'un octet dont le codage est indiqué au registre A. !
! BC2F H !
! 48175 D !
! _____!
!
!          PARAMETRES          ! INDICATEURS! REG. !
!
!          ENTREE          SORTIE ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! A -> codage de l'octet ! A -> numéro de ! oui !
!
!          ! l'encre !
!
!          !
!
!          !
!
!          !
!
!-----!
! NOTES:
!
=====

```

```

=====
! ROUTINE ! FONCTIONS: Fixe la couleur d'une encre. Si les deux !
!          ! couleurs fournies sont identiques, l'encre est fixe !
! BC32 H ! sinon elle clignote. !
! 48178 D ! !
! _____! !
!          PARAMETRES          ! INDICATEURS! REG. !
!          ENTREE              SORTIE    ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! A -> numéro de l'encre      !          ! oui      ! A !
!                              !          !          ! BC !
! B -> numéro 1ère couleur    !          !          ! DE !
!                              !          !          ! HL !
! C -> numéro 2ème couleur    !          !          !    !
!                              !          !          !    !
!-----!-----!-----!-----!
! NOTES: Voir exemple 1. !
!                              !
=====

```

```

=====

! ROUTINE ! FONCTIONS: Renseigne sur les couleurs d'une encre.  !

!           !                                                    !

! BC35  H !                                                    !

! 48181 D !                                                    !

! _____!_____!

!           PARAMETRES                ! INDICATEURS! REG. !

!           ENTREE                     SORTIE    ! AFFECTES. ! AFF. !

!-----!-----!-----!-----!

! A -> numéro de l'encre    ! B -> 1ère coul. !   oui   ! A   !

!                           !           !           ! DE  !

!                           ! C -> 2ème coul. !           ! HL  !

!                           !           !           !     !

!                           !           !           !     !

!                           !           !           !     !

!-----!

! NOTES:                                     !

!                                           !

=====

```



```

=====
! ROUTINE ! FONCTIONS: Fixe la couleur du bord. Les numéros des !
!          ! couleurs vont de 0 à 26. La période de clignotement !
! BC38 H ! peut être changée par la routine BC3EH.          !
! 48184 D !                                                  !
! _____! _____!

```

```

!          PARAMETRES          ! INDICATEURS! REG. !
!          ENTREE              SORTIE    ! AFFECTES. ! AFF. !

```

```

!-----!-----!-----!-----!

```

```

! B -> 1ère couleur      !          ! oui    ! A  !
!                      !          !          ! BC !
! C -> 2ème couleur      !          !          ! DE !
!                      !          !          ! HL !
!                      !          !          !    !
!                      !          !          !    !

```

```

!-----!

```

```

! NOTES: Voir exemple 1.          !
!                                !

```

```

=====

```

```

=====
! ROUTINE ! FONCTIONS: Renseigne sur la couleur du bord.      !
!          !                                                    !
! BC3B  H !                                                    !
! 48187 D !                                                    !
! _____!_____!_____!_____!
!
!          PARAMETRES          !INDICATEURS! REG. !
!
!          ENTREE          SORTIE    ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
!
!          ! B -> 1ère coul. ! Z=1 ; N=1 ! A  !
!
!          !          !          ! DE  !
!
!          ! C -> 2ème coul. ! reste à 0 ! HL !
!
!          !          !          !      !
!
!          !          !          !      !
!
!          !          !          !      !
!-----!
! NOTES:                                     !
!
!
=====

```

```

=====

! ROUTINE ! FONCTIONS: Fixe le temps pendant lequel apparait !
!
!      ! chacune des 2 couleurs des encres clignotantes. Il !
! BC3E H ! s'exprime en unité allant de 0 à 256. Chaque unité !
! 48190 D ! a une valeur de 1/50 s.
!
! _____!
!
!          PARAMETRES          !INDICATEURS! REG. !
!
!      ENTREE          SORTIE    ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! H -> temps 1ère couleur    !          ! non    !      !
!
!          !          !          !      !
! l -> temps 2ème couleur    !          !          !
!
!          !          !          !      !
!          !          !          !      !
!          !          !          !      !
!          !          !          !      !
!-----!
! NOTES:
!
!
=====

```

```

=====
! ROUTINE ! FONCTIONS: Renseigne sur le temps d'affichage des 2 !
!
!      ! couleurs des encres clignotantes. Il est donné en !
! BC41 H ! unités vallant chacune 1/50 s.                  !
! 48193 D !                                                  !
! _____! _____!
!
!              PARAMETRES              !INDICATEURS! REG. !
!
!      ENTREE              SORTIE      ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
!
!              ! H -> temps 1ère !      non      !      !
!
!              !      couleur      !              !      !
!
!              !              !              !      !
!
!              ! L -> temps 2ème !              !      !
!
!              !      couleur      !              !      !
!
!              !              !              !      !
!-----!
! NOTES:                                                  !
!
!                                                  !
=====

```

```

=====
! ROUTINE ! FONCTIONS: Remplissage avec une encre codée d'une !
!           ! zone rectangulaire de l'écran dont les limites sont !
! BC44 H ! données en coordonnées de caractère par rapport au !
! 48196 D ! coin haut à gauche (0,0). !
! _____!
!
!           PARAMETRES           ! INDICATEURS! REG. !
!
!           ENTREE           SORTIE ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! A -> code encres des pixels!           ! Z=1 ; N=1 ! A !
!
!           !           !           ! BC !
! H -> colonne gauche !           ! reste à 0 ! DE !
! D -> colonne droite !           !           ! HL !
! L -> ligne haut !           !           !
! E -> ligne bas !           !           !
!-----!
! NOTES: Le système ne corrige pas les coordonnées illégales. !
!
!
=====

```

```

=====

! ROUTINE ! FONCTIONS: Remplissage avec une encre codée d'une !
!           ! zone rectangulaire de l'écran dont les limites sont !
! BC47 H ! données à partir d'une adresse de base, d'une largeur !
! 48199 D ! en octets et d'une hauteur en lignes de pixels.      !
! _____!_____!
!
!           PARAMETRES                ! INDICATEURS! REG. !
!
!           ENTREE                      SORTIE      ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! C -> code encres des pixels!                ! Z=1 ; N=1 ! A      !
! HL -> adresse du coin haut !                !           ! BC      !
!           à gauche de la zone !                !           ! DE      !
! D -> largeur en octets      !                !           ! HL      !
! E -> hauteur en lignes de  !                !           !         !
!           pixels           !                !           !         !
!-----!
! NOTES: Le système ne corrige pas l'envoi de limites illégales. !
! Voir exemple 2.                                     !
=====

```

```

=====

! ROUTINE ! FONCTIONS: Transforme pour un caractère, les couleurs !
!
! de certains pixels dont l'encre codée est fournie, en !
! BC4A H ! d'autres couleurs dont le code d'encre sera indiqué. !
! 48202 D ! (coordonnées caractère / coin haut à gauche (0,0)). !
! _____!
!
! PARAMETRES ! INDICATEURS! REG. !
!
! ENTREE SORTIE ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! B -> code encres pixels ! ! Z=1 ; N=1 ! A !
! C -> nouveau code encres ! ! ! BC !
! ! ! ! DE !
! H -> colonne caractère ! ! ! HL !
! L -> ligne caractère ! ! ! !
! ! ! ! !
!-----!
! NOTES: Le système ne corrige pas des coordonnées illégales. !
!
=====

```

```

=====

! ROUTINE ! FONCTIONS: Scrolling de tout l'écran, de 8 lignes de !
!          ! pixel, soit un caractère. Celui-ci peut s'effectuer !
! BC4D H ! vers le bas ou vers le haut. Il sera indiqué, l'encre !
! 48205 D ! de la nouvelle ligne apparaissant à l'écran.      !
! _____!_____!
!
!          PARAMETRES          ! INDICATEURS! REG. !
!
!          ENTREE          SORTIE    ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! scrolling haut: B <> 0    !          ! Z=1 ; N=1 ! A  !
!
!          !          !          ! BC  !
! scrolling bas: B = 0    !          ! reste à 0 ! DE !
!
!          !          !          ! HL  !
! A -> code encres nouvelle !          !          !
!      ligne          !          !          !
!-----!
! NOTES:
!
!
=====

```



```

=====

! ROUTINE ! FONCTIONS: Conversion d'une matrice standard en une !

!           ! série de caches de pixels, appropriés au mode écran !

! BC53 H ! en cours. Ainsi chaque octet de la matrice sera écrit !

! 48211 D ! sur 4, 2, ou 1 octet selon que le mode est 0, 1 ou 2. !

! _____ !

!           PARAMETRES           ! INDICATEURS ! REG. !

!           ENTREE                SORTIE    ! AFFECTES. ! AFF. !

! .....! .....! .....! .....!

! HL -> adresse de la matrice!           ! oui    ! A    !

!           !           !           ! BC    !

! DE -> adresse de la zone où!           !       ! DE    !

! stocker les caches. !           !       ! HL    !

!           !           !           !       !

!           !           !           !       !

! .....!

! NOTES:                               !

!                               !

=====

```

```

=====
! ROUTINE ! FONCTIONS: Conversion de la matrice d'un caractère !
!          ! à l'écran, en une matrice standard de 8 octets.      !
! BC56 H !                                                         !
! 48214 D !                                                         !
! _____!_____!
!
!          PARAMETRES          ! INDICATEURS! REG. !
!
!          ENTREE              SORTIE      ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! A -> code encres des pixels!          ! tout à 0 ! A  !
!
!          !          !          ! BC !
! H -> colonne caractère  !          ! DE !
! L -> ligne  caractère  !          ! HL !
! DE -> adresse de stockage !          !      !
!          de la matrice  !          !      !
!-----!
! NOTES:
!
!
=====

```

```

! ROUTINE ! FONCTIONS: Fixe le mode d'écriture graphique . Voir !
!
! explications dans le paragraphe précédent. En cas de !
! BC59 H ! valeur illégale, le système calcule et utilise son !
! 48217 D ! modulo 3.
!
!
!

```

```

=====
! ROUTINE ! FONCTIONS: Place des pixels à l'écran en ignorant le !
!           ! mode d'écriture graphique. La position des pixels est !
! BC5C H ! indiquée par leur adresse.                               !
! 48220 D !                                                           !
! _____!_____!
!
!           PARAMETRES                ! INDICATEURS! REG. !
!
!           ENTREE                     SORTIE    ! AFFECTES. ! AFF. !
!-----|-----|-----|-----!
! HL -> adresse des pixels  !           ! tout à 0 ! A  !
!                           !           !         !     !
! B -> code encres des pixels!           !         !     !
!                           !           !         !     !
! C -> cache des pixels    !           !         !     !
!                           !           !         !     !
!-----|
! NOTES: Voir exemple 1.                                           !
!                                                                     !
=====

```

```

=====
! ROUTINE ! FONCTIONS: Traçage à l'écran d'une ligne horizontale. !
!
! Elle est tracée en fonction du code des encres fourni !
! BC5F H ! et du MEG. Les coordonnées de la ligne sont données !
! 48223 D ! par rapport au coin bas à gauche (0,0). !
! _____ !
!
! PARAMETRES ! INDICATEURS! REG. !
!
! ENTREE SORTIE ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! A -> code encres des pixels! ! Z=1 ; N=1 ! A !
! ! ! BC !
! BC ->coor. X fin de ligne ! ! reste à 0 ! DE !
! DE ->coor. X début de ligne! ! ! HL !
! HL ->coor. Y de la ligne ! ! ! !
! ! ! !
!-----!
! NOTES: Les coordonnées sont en pixels et dépendent donc du mode !
! Attention aux valeurs illégales. Voir exemple 2. !
=====

```

```

=====

! ROUTINE ! FONCTIONS: Traçage à l'écran d'une ligne verticale en !
!
! fonction d'une encre codée et du MEG. Les coordonnées !
! BC62 H ! seront exprimées en pixels, par rapport au coin bas à !
! 48226 D ! gauche (0,0), et en fonction du mode en cours.      !
! _____!
!
!                PARAMETRES                !INDICATEURS! REG. !
!
!                ENTREE                SORTIE    ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! A -> code encres des pixels!                ! Z=1 ; N=1 ! A  !
!
!                !                !                ! BC  !
! BC ->coord. Y fin de ligne !                ! reste à 0 ! DE  !
! DE ->coord. X de la ligne !                !                ! HL  !
! HL ->coord. Y début de ligne!                !                !
!
!                !                !                !
!-----!
! NOTES: Attention aux coordonnées illégales. Voir exemple 2.    !
!
!
=====

```

III - EXEMPLES D'UTILISATION:

Nous vous proposons 2 exemples d'utilisation de ces routines. Le premier permet d'afficher un petit personnage à l'écran. Après initialisation et mise en mode 0 de l'écran, des numéros de couleurs, stockés en 6550H, seront attribués aux encres. Les couleurs du bord seront données et l'écran effacé. Puis, après avoir fixé l'emplacement du personnage, les octets le composant seront envoyés à l'écran.

Les valeurs de ces octets codés ont été déterminées grace au tableau qui se trouve dans le paragraphe sur les généralités et seront stockées de 6500H à 653FH. L'adresse 6540H servira à conserver l'adresse du prochain octet de l'écran...

Le deuxième exemple consistera à colorier une zone de l'écran avec une valeur codée. Le mode utilisé sera le mode 1. Une fois la zone tracée, elle sera encadrée par des lignes continues. L'appui d'une touche provoquera alors un scrolling vers le bas de la moitié droite de la zone, puis un retour de celle-ci à sa position d'origine et pour terminer un scrolling vers le haut de cette moitié de zone...

=== EXEMPLE 1 ===

```
ORG 7000H

LOAD $

STO: EQU 6500H ;CODE ENCREs OCTETS

PLA: EQU 6540H

ENC: EQU 6550H ;COULEURS ENCREs
```


7000	CDFFB8		CALL	0BBFFH	;INITIALISATION
7003	215065	ENCRE:	LD	HL,ENC	
7006	7E	E1:	LD	A,(HL)	
7007	23		INC	HL	
7008	46		LD	B,(HL)	
7009	4E		LD	C,(HL)	
700A	E5		PUSH	HL	
700B	CD32BC		CALL	0BC32H	;COULEURS DES ENCRE
700E	E1		POP	HL	
700F	23		INC	HL	
7010	3E00		LD	A,0	
7012	BE		CP	(HL)	
7013	20F1		JR	NZ,E1	
7015	0600	BORD:	LD	B,0	
7017	0E00		LD	C,0	
7019	CD38BC		CALL	0BC38H	;COULEUR BORD
701C	3E00		LD	A,0	
701E	CD0EBC		CALL	0BC0EH	;MODE
7021	CD14BC		CALL	0BC14H	;CLS ECRAN
7024	2158C0		LD	HL,0C058H	;DEBUT AFFICHAGE

7027	224065		LD	(PLA),HL	
702A	1E00	AFF:	LD	E,0	
702C	DD210065		LD	IX,6500H	
7030	1600	A1:	LD	D,0	
7032	2A4065		LD	HL,(PLA)	
7035	E5		PUSH	HL	
7036	CD20BC		CALL	OBC20H	;ADRESSE OCTET DROIT
7039	224065		LD	(PLA),HL	
703C	E1		POP	HL	
703D	DD4600	A2:	LD	B,(IX+0)	
7040	0EFF		LD	C,0FFH	;CACHE
7042	CD5CBC		CALL	OBC5CH	;AFFICHAGE PIXELS
7045	CD26BC		CALL	OBC26H	;ADRESSE OCTET DESSOUS
7048	DD23		INC	IX	
704A	14		INC	D	
704B	7A		LD	A,D	
704C	FE10		CP	10H	
704E	20ED		JR	NZ,A2	
7050	1C		INC	E	
7051	7B		LD	A,E	

7052	FE04	CP	4	
7054	20DA	JR	NZ,A1	
7056	CD06BB	CALL	0BB06H	;ATTENTE FRAPPE TOUCHE
7059	C9	RET		

END

--- LISTING DES DONNEES ---

6500: 00 00 00 10 10 64 64 64

6508: 64 10 10 00 00 50 F0 00

6510: 00 10 64 CC CC 4C CC CC

6518: C4 C0 C8 64 B0 A0 00 00

6520: 00 20 98 CC CC 8C CC CC

6528: C8 C0 C4 98 70 50 00 00

6530: 00 00 00 20 20 98 98 98

6538: 98 20 20 00 00 A0 F0 00

6550: 00 00 01 06 02 02 03 16

6558: 04 12 05 0F 00 00 00 00

=== EXEMPLE 2 ===

ORG 7000H

LOAD \$

X1: EQU 100 ;BORD GAUCHE

X2: EQU 218 ;BORD DROIT

Y1: EQU 50 ;BORD BAS

Y2: EQU 150 ;BORD HAUT

7000	CDFFB8	CALL	08BFFH	;INITIALISATION
7003	3E01	LD	A,1	
7005	CD0EBC	CALL	0BC0EH	;MODE 1
7008	116800	LD	DE,X1+4	
700B	219200	LD	HL,Y2-4	
700E	CD1DBC	CALL	0BC1DH	;CALCUL ADRESSE DEBUT ZONE
7011	0EA6	LD	C,0A6H	
7013	161C	LD	D,28	
7015	1E5D	LD	E,93	
7017	CD47BC	CALL	0BC47H	;REMPLISSAGE ZONE

701A	3E0F	LD	A,0FH	
701C	116400	LD	DE,X1	
701F	01DA00	LD	BC,X2	
7022	219600	LD	HL,Y2	
7025	CD5FBC	CALL	OBC5FH	;LIGNE HORIZONTALE
7028	3E0F	LD	A,0FH	
702A	116400	LD	DE,X1	
702D	01DA00	LD	BC,X2	
7030	213200	LD	HL,Y1	
7033	CD5FBC	CALL	OBC5FH	;LIGNE HORIZONTALE
7036	3E0F	LD	A,0FH	
7038	116400	LD	DE,X1	
703B	213200	LD	HL,Y1	
703E	019600	LD	BC,Y2	
7041	CD62BC	CALL	OBC62H	;LIGNE VERTICALE
7044	3E0F	LD	A,0FH	
7046	11DA00	LD	DE,X2	
7049	213200	LD	HL,Y1	
704C	019600	LD	BC,Y2	

704F	CD62BC		CALL	OBC62H	;LIGNE VERTICALE
7052	CD068B		CALL	OBB06H	;ATTENTE FRAPPE TOUCHE
7055	3E00		LD	A,0	
7057	320071		LD	(7100H),A	
705A	CD6B70		CALL	SCROL	
705D	CD068B		CALL	OBB06H	;ATTENTE FRAPPE TOUCHE
7060	3E01		LD	A,1	
7062	320071		LD	(7100H),A	
7065	CD6B70		CALL	SCROL	
7068	CD068B		CALL	OBB06H	;ATTENTE FRAPPE TOUCHE
706B	0600	SCROL:	LD	B,0	
706D	04	SC1:	INC	B	
706E	C5		PUSH	BC	
706F	210071		LD	HL,7100H	
7072	46		LD	B,(HL)	
7073	3E00		LD	A,0	
7075	2614		LD	H,20	
7077	161E		LD	D,30	
7079	2E01		LD	L,1	
707B	1E18		LD	E,24	

707D	CD50BC	CALL	OBC50H	;SCROLLING
7080	C1	POP	BC	
7081	3E05	LD	A,5	
7083	B8	CP	B	
7084	20E7	JR	NZ,SC1	
7086	C9	RET		

END

CHAPITRE 8 -

L'UNITE DE CASSETTE

I - GENERALITES:

L'amstrad offre à l'utilisateur deux vitesses de sauvegarde des fichiers (programmes, données, pages écran, etc ...).

L'utilisateur peut utiliser, soit la vitesse lente (1000 bauds), soit la vitesse rapide (2000 bauds); dans le premier cas il est assuré de la fiabilité du transfert des données, tandis que dans le deuxième cas, les risques d'erreurs sont plus grands.

A la lecture d'un fichier, l'amstrad détermine lui-même la vitesse de lecture, libérant ainsi l'utilisateur du souvenir de la vitesse de sauvegarde de son fichier.

A 1000 bauds, le lecteur de cassette écrit ou lit 125 caractères par seconde, ce qui permet d'obtenir des transferts de données de 250 caractères pour 2000 bauds.

Par ailleurs (voir routines du système d'exploitation), il est possible de fixer des vitesses différentes.

II - ORGANISATION D'UN FICHIER:

Un fichier fait apparaître deux types d'enregistrement; le premier regroupe un certain nombre d'informations relatives au fichier, et le second représente les données du fichier.

Toutefois un fichier n'est pas écrit en un seul bloc; l'amstrad, quant à lui, coupe un fichier en blocs de 2k octets (2048 octets). Devant chaque bloc le système écrit le 1er type d'enregistrement; celui-ci, d'une longueur de 64 octets, indique des informations relatives au fichier, ainsi que des informations relatives au bloc de données qui suit (No de bloc, longueur des données). En début de chaque fichier, le système écrit un signal qui permet à l'amstrad de déterminer le début de celui-ci sur la cassette.

L'ENREGISTREMENT EN-TETE D'UN FICHIER:

Cet enregistrement est très important; en effet celui-ci contient toutes les informations relatives au fichier, ainsi que pour le bloc en cours; les renseignements données sont —>

OCTETS	FONCTIONS	ADRESSES
0 a 15	Nom du fichier	&B807
16	Numéro du bloc	&B817
17	Dernier bloc (si <> 0)	&B818
18	Type de fichier:	&B819

	bit 0 -> protection (si = 1)	
	bits 1 a 3 -> basic (si = 0)	
	-> binaire (si = 1)	
	-> écran (si = 2)	
	-> ASCII (si = 3)	
	bits 4 a 7 -> non utilisé (vers.)	
19 et 20	Longueur des données	&B81A
21 et 22	Adresse début des données	&B81C
23	Premier bloc (si <> 1)	&B81E
24 et 25	Longueur logique du fichier	&B820
26 et 27	Adresse du point d'entrée pour un programme en code machine.	&B822
28 a 63	Non utilisé (commentaires)	&B824

Cet enregistrement est long de 64 octets et se positionne en &B807 au moment de sa lecture; toutefois il est important à noter qu'une fois que le système l'a traité, la zone est réinitialisée. A partir de cet enregistrement nous pouvons donc faire un certain nombre de traitement, et notamment des duplications de logiciels personnels, ou du commerce, avec l'autorisation de l'éditeur. A ce sujet, vous trouverez en annexe, un programme basic effectuant ce type de traitement; ce logiciel permet, outre les duplications, de lire seulement le catalogue, et d'en garder une trace écrite.

Il existe une importante distinction entre les enregistrements de tête, et les enregistrements de données;

Pour que le système de l'amstrad puisse faire la différence entre en-tête et donnée, celui-ci écrit les enregistrements de tête avec un caractère de synchronisation de &2C, et écrit les données avec un caractère de synchronisation de &16.

III - ROUTINES D'UTILISATION DE L'UNITE CASSETTE:

Un certain nombre de routines peuvent être utilisées dans le but de créer et de lire des fichiers standards, ou des fichiers ayant une organisation particulière.

Dans les pages qui vont suivre, il est présenté un catalogue de routines, à raison de trois routines par pages; celles-ci sont classées comme suit:

- (1) routines de lecture
- (2) routines d'écriture
- (3) routine "catalogue"

Pour chaque routine est indiquée, son adresse d'exécution (H et D), sa fonction, les paramètres éventuels en entrée (par le biais des registres), les paramètres en sortie (mis dans les registres), les registres et indicateurs affectés après exécution de la routine, et des observations éventuelles sur le déroulement de l'exécution de celle-ci.

== NOTES SUR LES REGISTRES ET INDICATEURS UTILISES ==

- REGISTRES -

- INDICATEURS -

F: registre d'état. | H: demi-retenue

IX: registre d'index X. | O: débordement (overflow)

IY: registre d'index Y. |

LEGENDE: OK - exécution correcte / ER - erreur / ES: escape

FF - fin de fichier / NO - non ouvert / DO: déjà ouvert

LC - lecture en cours / EU - en utilisation.

=====

! ROUTINE ! FONCTIONS: initialisation de l'unité cassette. !

! ! initialise la zone de réception de l'en-tete. !

! BC65 H ! autorise l'affichage des messages en lecture et en !

! 48229 D ! écriture; vitesse d'écriture rapide. !

! _____ !

! PARAMETRES ! INDICATEURS! REG. !

! ENTREE SORTIE ! AFFECTES. ! AFF. !

! !

! aucun ! aucun ! oui ! DE !

! ! ! HL !

! ! ! A !

! ! ! !

! ! ! !

! ! ! !

! !

! NOTES: néant !

! !

=====

```

=====

! ROUTINE ! FONCTIONS: autorise ou non l'affichage des messages !

!           ! en lecture et en écriture.                             !

! BC6B H ! voir en notes les messages possibles.                   !

! 48235 D !                                                            !

! _____!_____!

!                               PARAMETRES                          !INDICATEURS! REG. !

!           ENTREE                SORTIE      ! AFFECTES. ! AFF. !

!-----!-----!-----!-----!

! A = 0 -> messages autorisés! aucun          ! oui        ! non !

! A <> 0 -> non affichage des!                !            !    !

!           messages          !                !            !    !

!                               !                !            !    !

!                               !                !            !    !

!                               !                !            !    !

!-----!-----!-----!-----!

! NOTES: press play .../ found .../ loading .../ read error ... !

! rewind tape / press rec .../ saving .../ write error ...      !

=====

```

```

=====

! ROUTINE ! FONCTIONS: démarrage du moteur !
!
!
! BC6E H !
! 48238 D !
! _____!
!
! PARAMETRES ! INDICATEURS! REG. !
!
! ENTREE SORTIE ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! aucun ! A = 0 -> état du! OK:C=1 ! A !
!
! moteur ! ES:C=0 !
!
!
!
!
!
!
!-----!
! NOTES: l'état du moteur sert pour la routine de restauration du !
! moteur. !
=====

```

=====

! ROUTINE ! FONCTIONS: arrêt du moteur cassette. !

! ! !

! BC71 H ! !

! 48241 D ! !

! _____ ! _____ !

! PARAMETRES ! INDICATEURS! REG. !

! ENTREE SORTIE ! AFFECTES. ! AFF. !

! -----! -----! -----! -----!

! aucun ! OK: A = 10H. ! OK:C=1 ! A !

! ! si déjà arrêté ! ES:C=0 ! !

! ! A = 0. ! ! !

! ! (A: état moteur)! ! !

! ! ! ! !

! ! ! ! !

! -----! -----! -----! -----!

! NOTES: l'état du moteur sert pour la routine de restauration du !

! moteur. !

=====


```

=====

! ROUTINE ! FONCTIONS: restauration état moteur suivant le contenu!

!           ! de A; si moteur en marche on l'arrête et inversement. !

! BC74  H !                                     !

! 48244 D !                                     !

! _____! _____!

!           PARAMETRES           !INDICATEURS! REG. !

!           ENTREE           SORTIE   ! AFFECTES. ! AFF. !

!-----!-----!-----!-----!

! si A = 0 et moteur tourne ! ES ou arrêt et ! OK:C=1   ! A   !

! --> arrêt moteur.         ! A=10H avant   ! ES:C=0   !     !

! si A = 10H et moteur arrête! exécution     !         !     !

! --> mise en marche moteur. ! --> A = 10H   !         !     !

!                             !                 !         !     !

!                             !                 !         !     !

!-----!-----!-----!-----!

! NOTES: A peut être renseigné à partir des routines de mise en   !

! en marche, et d'arrêt du moteur.                                !

=====

```

```

=====

! ROUTINE ! FONCTIONS: ouverture d'un fichier et lecture du      !
!          ! premier bloc.                                         !
! BC77 H !                                                         !
! 48247 D !                                                         !
! _____! _____!                                         !

!          PARAMETRES          ! INDICATEURS! REG. !
!          ENTREE              SORTIE    ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! B: longueur nom fichier    ! BC: long nom    ! OK:C=1/Z=0! IX,BC!
! HL: adresse nom fichier    ! DE: adr déb fich!    N=1/S=1! DE,HL!
! DE: adresse buffer de 2K    ! HL: adr buffer    ! DO:tous à ! A.    !
!      pour réception bloc.    ! A: type de fich    ! zéro.    !      !
! si nom fichier = 0 -->      !          ! ES:C=0/Z=1! ES:    !
! lecture du 1er fich trouvé !          ! N=0/S=0! IX,A    !
!-----!
! NOTES: nom fichier < 16 octets , et suivi d'un 0 pour détection !
! fin de texte par le système.                                     !
=====

```

```

=====

! ROUTINE ! FONCTIONS: lecture d'un enregistrement de données      !
!
!          ! (caractère de synchro: 16H), ou d'un enregistrement    !
! BCA1 H ! en-tête (caractère de synchro: 2CH).                      !
! 48289 D !                                                           !
! _____! _____!                                           !
!
!          PARAMETRES          ! INDICATEURS! REG. !
!
!          ENTREE              SORTIE      ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! HL: adresse des données      ! si erreur de      ! OK:C=1      ! IX,A !
! DE: longueur des données     ! lecture A =      ! ES:C=0      ! BC,D !
! A: caractère de synchro      ! type d'erreur    ! ER:tous à   ! HL   !
!
!                               !          ! zéro.      !      !
!
!                               !          !          !      !
!
!                               !          !          !      !
!-----!-----!-----!-----!

! NOTES: il est possible de lire le catalogue d'un fichier même    !
! protégé (voir programme de duplication en annexe).                !
=====

```

```

=====

! ROUTINE ! FONCTIONS: fermeture d'un fichier lu sur cassette.  !

!           !                                                    !

! BC7A  H !                                                    !

! 48250 D !                                                    !

! _____! _____!

!                                     PARAMETRES                !INDICATEURS! REG. !

!          ENTREE                SORTIE          ! AFFECTES. ! AFF. !

! -----! -----! -----! -----!

! aucun                ! si OK -> A=FFH ! OK:0=0      !OK:A, !

!                   ! si NO -> A=0    ! C=N=H=S=1 !  HL !

!                   !                   !           !      !

!                   !                   ! NO:0=1    !NO:A !

!                   !                   ! C=N=H=S=0 !      !

!                   !                   !           !      !

! -----! -----! -----! -----!

! NOTES: il est important de fermer ses fichiers, sinon il ne  !

! pourront etre lus par la suite.                                !

=====

```

```

=====
! ROUTINE ! FONCTIONS: lecture d'un caractère sur la cassette. !
!          !                                                    !
! BC80  H !                                                    !
! 48256 D !                                                    !
! _____! _____!
!
!          PARAMETRES          ! INDICATEURS! REG. !
!
!          ENTREE              SORTIE      ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! aucun                ! A = caractère lu! OK:C=1/Z=0! IX,A !
!
!                      !          ! FF:C=Z=1 !      !
!
!                      !          ! ES:C=0/Z=1!      !
!
!                      !          !          !      !
!
!                      !          !          !      !
!
!                      !          !          !      !
!-----!
! NOTES:
!
=====

```

```

=====

! ROUTINE ! FONCTIONS: lecture d'un fichier, à partir du      !
!          ! deuxième bloc.                                     !
! BC83  H !                                                    !
! 48259 D !                                                    !
! _____! _____!                                     !
!
!          PARAMETRES                ! INDICATEURS! REG. !
!
!          ENTREE                    SORTIE    ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! HL: adresse de réception  ! HL: adresse  ! OK:C=1/Z=0! C,A !
!   du fichier en RAM      ! du fichier  ! NO:tous à ! HL  !
!
!                          ! si OK.    ! zéro.    !
!
!                          !          ! ES:C=0/Z=1!
!
!                          !          !
!
!                          !          !
!-----!
! NOTES: cette routine doit etre exécutée après la routine BC77H. !
!
=====

```

```

=====
! ROUTINE ! FONCTIONS: fixe la vitesse d'écriture sur cassette. !
!           !
! BC68 H !
! 48232 D !
! _____!
!
!           PARAMETRES           ! INDICATEURS! REG. !
!
!           ENTREE           SORTIE   ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! HL:vitesse (micro-secondes)! aucun           ! oui           ! A,HL !
! A:compensation ( " )           !           !           !
! si 1000 bauds -> A=25           !           !           !
!           HL=333           !           !           !
! si 2000 bauds -> A=50           !           !           !
!           HL=167           !           !           !
!-----!
! NOTES: la vitesse peut etre différente de 1000 et 2000 bauds. !
!
=====

```

```

=====

! ROUTINE ! FONCTIONS: ouverture d'un fichier en sortie, pour      !
!           ! permettre l'écriture de celui-ci sur la cassette.    !
! BC8C  H !                                                         !
! 48268 D !                                                         !
! _____! _____!                                         !
!
!           PARAMETRES                !INDICATEURS! REG. !
!
!           ENTREE                    SORTIE    ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! B: longueur du nom de fich ! aucun          ! OK:C=1/Z=0! A,C  !
! HL: adresse nom de fichier !                ! ES:C=0/Z=1! DE   !
! DE: adresse du buffer de 2K!                ! EU:tous à ! HL   !
!
!           !                !   zéro.  ! IX   !
!
!           !                !           !       !
!
!           !                !           !       !
!-----!-----!-----!-----!
! NOTES:                                     !
!
!                                           !
=====

```



```

=====

! ROUTINE ! FONCTIONS: écriture du dernier bloc, et fermeture !
!
! du fichier. !
!
! BC8F H !
!
! 48271 D !
!
! _____!
!
! PARAMETRES ! INDICATEURS! REG. !
!
! ENTREE SORTIE ! AFFECTES. ! AFF. !
!
! .....! .....! .....! .....!
!
! aucun ! OK: DE = adresse! OK:C=1 ! A,BC !
!
! ! buffer de 2K! Z=S=0 ! DE !
!
! ! ! NO:S=1 ! !
!
! ! NO: A = FFH ! C=Z=0 ! NO:A !
!
! ! ! ES:Z=1 ! !
!
! ! ! C=S=0 ! !
!
! .....!
!
! NOTES: si la touche ESC est appuyée pendant l'écriture du !
! dernier bloc, le fichier ne sera pas fermé. !
=====

```

```

=====
! ROUTINE ! FONCTIONS: écriture d'un caractère. !
!          !                                     !
! BC95  H !                                     !
! 48277 D !                                     !
! _____!_____!
!
!          PARAMETRES          ! INDICATEURS! REG. !
!          ENTREE              SORTIE  ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! A = caractère à écrire.  ! aucun      ! OK:C=1/Z=0! A,IX !
!
!
!
!
!
!
!
!
! NO:C=Z=0  !
!
!
!
! ES:C=0/Z=1!
!
!
!
!-----!
! NOTES:
!
=====

```

```

=====

! ROUTINE ! FONCTIONS: écriture d'un fichier en sortie.      !

!           !                                                  !

! BC98 H !                                                  !

! 48280 D !                                                  !

! _____! _____!

!           PARAMETRES           ! INDICATEURS! REG. !

!           ENTREE           SORTIE           ! AFFECTES. ! AFF. !

!-----!-----!-----!-----!

! HL: adresse des données      ! aucun           ! OK:C=S=1 ! A,IX !

! DE: longueur des données      !                 ! Z=N=0 ! BC,DE!

! BC: adresse exécution si      !                 ! NO:tous à ! HL. !

!      fichier hexa (assembl) !                 ! zéro !      !

! A: type de fichier (voir      !                 ! ES:C=S=0 ! NO:IX!

!      enreg. en-tete)          !                 ! Z=N=1 ! inch.!

!-----!

! NOTES: seuls les blocs 1 à l'avant-dernier sont écrits.      !

!           !                                                  !

=====

```

```

=====

! ROUTINE ! FONCTIONS: écriture d'un enregistrement particulier. !

!           !                                     !

! BC9E  H !                                     !

! 48286 D !                                     !

! _____! _____!

!           PARAMETRES           ! INDICATEURS! REG. !

!           ENTREE           SORTIE   ! AFFECTES. ! AFF. !

!-----!-----!-----!-----!

! HL: adresse enregistrement ! OK:A=0           ! OK:C=1/H=0! A,D  !

! DE: longueur enregistrement! ER/ES:A=code   ! ER/ES:C=0 ! IX,BC!

! A: caractère de synchro.  ! erreur.   ! H=1 ! HL.  !

!           !           !           !           !

! si longueur donnée = 0 -> ! si A=1 -> probl !           !ER/ES:!

! écrit 64K RAM.           ! de transfert.  !           !A seul!

!-----!

! NOTES: le caractère de synchro peut etre différent, mais il doit!

! etre repris pour la lecture du fichier ainsi créé.           !

=====

```

```

=====
! ROUTINE ! FONCTIONS: affichage des informations relatives à      !
!          ! un fichier.                                           !
! BC9B H !                                                         !
! 48283 D !                                                         !
! _____!
!
!          PARAMETRES          ! INDICATEURS! REG. !
!
!          ENTREE              SORTIE      ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! DE: adresse du buffer de  ! aucun          ! OK:C=1/Z=0! A      !
! 2K.                      !                !          ! BC      !
!                          !                ! LC:C=Z=0  ! HL      !
!                          !                !          ! IX      !
!                          !                ! ER:C=0/Z=1!          !
!                          !                !          !          !
!-----!
! NOTES: affiche le nom du fichier, le numéro du bloc lu, et le  !
! type de fichier: & ' * $ % (voir signification ci-après).      !
=====

```

PRECISIONS SUR LE TYPE DE FICHIER:

si & → fichier binaire non protégé

si ' → fichier binaire protégé

si * → fichier texte (type ASCII)

si \$ → fichier basic non protégé

si % → fichier basic protégé

EXEMPLES D'UTILISATION DES ROUTINES CASSETTE:

A) LECTURE ET ECRITURE D'UN FICHIER STANDARD:

(1) lecture d'un fichier:

CALL &BC65 → initialisation gestion cassette

LD B,ln → ln: longueur du nom de fichier

LD HL,an → an: adresse du nom de fichier

LD DE,ab → ab: adresse du buffer de 2K

CALL &BC77 → ouverture et lecture du 1er bloc

LD HL,af → adresse de réception du fichier

CALL &BC83 → lecture du 2ème au dernier bloc

CALL &BC7A → fermeture complète du fichier

(2) écriture d'un fichier:

CALL &BC65 → initialisation gestion cassette
LD B,ln → ln: longueur du nom de fichier
LD HL,an → an: adresse du nom de fichier
LD DE,ab → ab: adresse du buffer de 2K
CALL &BC8C → ouverture du fichier en sortie
LD HL,ad → ad: adresse de début des données
LD DE,ld → ld: longueur de la zone données
LD BC,ae → ae: adresse d'exécution (assemb)
LD A,tf → tf: type de fichier à écrire
CALL &BC98 → écriture fichier (1er à AV dern)
CALL &BC8F → écriture dernier bloc et ferme

B) LECTURE ET ECRITURE D'UN ENREGISTREMENT:

(1) lecture d'un enregistrement particulier:

LD A,csH → cs: caractère de synchronisation
LD DE,le → le: longueur de l'enregistrement
LD HL,ae → ae: adresse de l'enregistrement
CALL &BCA1 → lecture de l'enregistrement

si cs = 2C → lecture d'un enregist. en-tête

'le' = 40H (64 octets).

si cs = 16 → lecture d'un enregist. données

(2) écriture d'un enregistrement particulier:

LD A,cs → cs: caractère de synchronisation

LD DE,le → le: longueur de l'enregistrement

LD HL,ae → ae: adresse de l'enregistrement

CALL &BC9E → écriture de l'enregistrement

si cs = 2C → écriture d'un enregist. en-tête

'le' = 40H (64 octets)

si cs = 16 → écriture d'un enregist. données

CHAPITRE 9 - ROUTINES "SYSTEMES"

I - GENERALITES:

Nous avons évoqué, au chapitre 2, les routines "RESTARTS" que le système utilise abondamment pour le passage RAM $\longleftrightarrow$ ROM.

A l'initialisation de l'amstrad, le système plante ces routines en RAM à partir de la ROM "basse". Ces routines étant essentiellement utilisées par le système, un certain nombre de routines "systèmes" est mis à la disposition de l'utilisateur;

Comme pour les "RESTARTS", ces routines sont obtenues à partir de la ROM "basse", et sont implantées dans le haut de la RAM (de &B900 à &BA5E). Grâce à celles-ci, l'utilisateur peut effectuer des connections en ROM, et peut également procéder à des transferts de blocs d'octets, de ROM en RAM ou de RAM en RAM. Il est bien entendu fort recommandé ne pas se servir de cette zone pour y planter des programmes ou données, sous peine de planter le système; en effet ces routines mises à la disposition de l'utilisateur sont également utilisées par le système. Il faut d'ailleurs savoir qu'outre ces routines dites "systèmes", la zone &B900 à &BDF1 est strictement réservée à l'implantation d'instructions d'appel de routines diverses (affichage, cassette, clavier, écran, etc ...).

Celles-ci sont largement traitées au fil des chapitres de cet ouvrage, et font toutes appel à des routines situées dans le système d'exploitation (ROM "basse").

Pour en revenir aux routines qui nous préoccupent dans l'immédiat, vous en trouverez dans les pages qui vont suivre, quelques une qui ont pour but la connection en ROM, et le transfert de données.

II - ROUTINES UTILISEES:

```
=====
! ROUTINE ! FONCTIONS: permet la lecture ou l'utilisation de      !
!          ! routines en ROM haute (&C000 à &FFFF).                !
! B900 H !                                                            !
! 47360 D !                                                            !
! _____!_____!
!
!          PARAMETRES          !INDICATEURS! REG. !
!
!          ENTREE              SORTIE      ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! Néant          ! A = état de la ! Aucun      ! A      !
!
!          ! ROM avant !          !          !
!          ! activation. !          !          !
!          !          !          !          !
!          !          !          !          !
!          !          !          !          !
!          !          !          !          !
!-----!-----!-----!-----!
! NOTES:
!
!_____!
=====
```

```

=====

! ROUTINE ! FONCTIONS: Désactive la ROM haute !
!           !                                     !
! B903 H !                                     !
! 47363 D !                                     !
! _____!                                     !

!           PARAMETRES           ! INDICATEURS! REG. !
!           ENTREE           SORTIE ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!

! Néant           ! A = état           ! Aucun           ! A           !
!           ! précédent           !           !           !
!           ! de la ROM           !           !           !
!           !           !           !           !
!           !           !           !           !
!           !           !           !           !
!           !           !           !           !
!-----!

! NOTES:           !
!           !
=====

```

=====

! ROUTINE ! FONCTIONS: Autorise l'accès en ROM basse, soit de !

! ! 0 à &3FFF. !

! B906 H !

! 47366 D !

! _____ !

! PARAMETRES ! INDICATEURS! REG. !

! ENTREE SORTIE ! AFFECTES. ! AFF. !

! !

! Néant ! A = état ! Aucun ! A !

! précédent ! !

! de la ROM ! !

! ! !

! ! !

! ! !

! !

! NOTES: !

! !

=====

```

=====

! ROUTINE ! FONCTIONS: désactive la ROM basse. !

! ! !

! B909 H ! !

! 47369 D ! !

! _____ !

!
!          PARAMETRES          ! INDICATEURS! REG. !
!
!      ENTREE          SORTIE    ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! Néant              ! A = état    ! Aucun    ! A    !
!
!                  ! précédent !          !      !
!                  ! de la ROM !          !      !
!                  !          !          !      !
!                  !          !          !      !
!                  !          !          !      !
!                  !          !          !      !
!-----!-----!-----!-----!
! NOTES: !
! ! !
=====

```

=====

! ROUTINE ! FONCTIONS: restaure l'état de
! la ROM suivant un

! état précédemment sauvegardé. !

! B90C H ! !

! 47372 D ! !

! _____ ! _____ !

! PARAMETRES ! INDICATEURS! REG. !

! ENTREE SORTIE ! AFFECTES. ! AFF. !

! ----- ! ----- ! ----- ! ----- !

! A = état précédent de la ! Aucun ! oui ! A !

! ROM haute ou basse. ! ! ! !

! ! ! ! !

! ! ! ! !

! ! ! ! !

! ! ! ! !

! ----- !

! NOTES: l'état précédent peut provenir des routines d'activation !

! ou de désactivation de la ROM haute ou basse. !

=====

```

=====

! ROUTINE ! FONCTIONS: transfert "ascendant" d'un bloc d'octets, !
!          ! suivant l'instruction assembleur LDIR.          !
! B91B H !                                                    !
! 47387 D !                                                    !
! _____!_____!
!
!          PARAMETRES          ! INDICATEURS! REG. !
!
!          ENTREE              SORTIE      ! AFFECTES. ! AFF. !
!-----!-----!-----!-----!
! BC: nombre d'octets à      ! Aucun      ! Aucun      ! BC !
! transférer.                !          !          ! DE !
! DE: adresse de début de    !          !          ! HL !
! réception du bloc.         !          !          ! A  !
! HL: adresse de départ du   !          !          !    !
! bloc à transférer.         !          !          !    !
!-----!
! NOTES: voir principe de fonctionnement à la fin du chapitre. !
!
=====

```


DIFFERENCE ENTRE LA ROUTINE &B91B (LDIR) ET LA ROUTINE &B91E (LDDR)

1) ROUTINE DE TRANSFERT &B91B:

Cette routine lit et transfère un groupe d'octets, suivant une méthode "ascendante"; c'est à dire qu'après la lecture d'un octet et son transfert, les adresses en HL et DE sont incrémentées de 1, et le registre BC est décrémenté de 1; il en est ainsi jusqu'à ce que le contenu du registre BC soit nul.

2) ROUTINE DE TRANSFERT &B91E:

Cette routine lit et transfère un groupe d'octets, suivant une méthode "descendante"; c'est à dire qu'après la lecture d'un octet et son transfert, les adresses en HL et DE sont décrémentées de 1, et le registre BC est décrémenté de 1; il en est ainsi jusqu'à ce que le contenu du registre BC soit nul.

```

EXEMPLE: BC=4 / HL=10 / DE=30  et  01 02 03 04 05 06 07 08 09 10
-----  --  (10)
(a) routine &B91B:

avant: 00 00 00 00 00 00 00 00 00 00  après: 00 00 00 00 05 06 07 08
-----  (30)  -----  (30)
(b) routine &B91E:

avant: 00 00 00 00 00 00 00 00 00 00  après: 00 02 03 04 05 00 00 00
-----  (30)  -----  (30)

```

ANNEXE 1 -

TABLES INVERSES POUR L'ASSEMBLEUR Z80:

Dans tous les manuels d'assembleur pour le Z80, vous trouverez des tables vous présentant une liste alphabétique des instructions de ce langage assembleur; ces tables sont utiles quand on écrit des programmes en langage assembleur.

Par ailleurs, pour ceux qui ne possèdent pas de désassembleur, il peut s'avérer intéressant de procéder au désassemblage de petites routines assembleur; pour cela il faut, à l'aide du code hexa, rechercher l'instruction assembleur correspondante; cette recherche peut être fastidieuse si on utilise la liste classique, où les instructions sont classées par ordre alphabétique, et non par classement sur le code hexa.

Pour pallier à ce manque, je vous propose d'utiliser des tables inverses; vous trouverez, dans les pages suivantes, des tables pour les instructions assembleurs dont le code hexa est sur 1 octet, des tables pour les codes instructions sur 2 octets, et des tables dont le code instruction est sur 3 octets. En haut et à droite de chaque case des tables, est inscrit un chiffre qui indique le nombre d'octet de l'instruction complète. Dans les instructions vous trouverez les littéraux n, nn, et e. n correspond à une donnée sur 1 octet; nn correspond à une donnée sur 2 octets; d et e sont des déplacements relatifs sur 1 octet.

Le premier digit d'un octet du code instruction est placé sur les lignes de chaque table, tandis que le deuxième digit est placé sur les colonnes de chaque table (de 0 à F pour les 2 digits). Pour les instructions sur 2 octets, il s'agira du deuxième octet, le 1er octet étant fixe; Pour les instructions sur 3 octets, il s'agira du 4ème octet, les premier et deuxième octets étant fixes.

INSTRUCTIONS DONT LE CODE EST SUR 1 OCTET (1ère partie)

	0	1	2	3	4	5	6	7
0 NOP	1 LD 'BC,nn	3 LD '(BC),A	1 INC 'BC	1 INC 'B	1 DEC 'B	1 LD 'B,n	2 RLCA	1
1 DJNZ 'e	2 LD 'DE,nn	3 LD '(DE),A	1 INC 'DE	1 INC 'D	1 DEC 'D	1 LD 'D,n	2 RLA	1
2 JR 'NZ,C	2 LD 'HL,nn	3 LD '(nn),HL	3 INC 'HL	1 INC 'H	1 DEC 'H	1 LD 'H,n	2 DAA	1
3 JR 'NC,e	2 LD 'SP,nn	3 LD '(nn),A	3 INC 'SP	1 INC '(HL)	1 DEC '(HL)	1 LD '(HL),n	2 SCF	1
4 LD 'B,B	1 LD 'B,C	1 LD 'B,D	1 LD 'B,E	1 LD 'B,H	1 LD 'B,L	1 LD 'B,(HL)	1 LD 'B,A	1
5 LD 'D,B	1 LD 'D,C	1 LD 'D,D	1 LD 'D,E	1 LD 'D,H	1 LD 'D,L	1 LD 'D,(HL)	1 LD 'D,A	1
6 LD 'H,B	1 LD 'H,C	1 LD 'H,D	1 LD 'H,E	1 LD 'H,H	1 LD 'H,L	1 LD 'H,(HL)	1 LD 'H,A	1
7 LD '(HL),B	1 LD '(HL),C	1 LD '(HL),D	1 LD '(HL),E	1 LD '(HL),H	1 LD '(HL),L	1 HALT	1 LD '(HL),A	1
8 ADD 'A,B	1 ADD 'A,C	1 ADD 'A,D	1 ADD 'A,E	1 ADD 'A,H	1 ADD 'A,L	1 ADD 'A,(HL)	1 ADD 'A,A	1
9 SUB 'B	1 SUB 'C	1 SUB 'D	1 SUB 'E	1 SUB 'H	1 SUB 'L	1 SUB '(HL)	1 SUB 'A	1
A AND 'B	1 AND 'C	1 AND 'D	1 AND 'E	1 AND 'H	1 AND 'L	1 AND '(HL)	1 AND 'A	1
B OR 'B	1 OR 'C	1 OR 'D	1 OR 'E	1 OR 'H	1 OR 'L	1 OR '(HL)	1 OR 'A	1
C RET 'NZ	1 POP 'BC	1 JP 'NZ,nn	3 JP 'nn	3 CALL 'NZ,nn	3 PUSH 'BC	1 ADD 'A,n	2 RST '00H	1
D RET 'NC	1 POP 'DE	1 JP 'NC,nn	3 OUT '(n),A	2 CALL 'NC,nn	3 PUSH 'DE	1 SUB 'n	2 RST '10H	1
E RET 'PO	1 POP 'HL	1 JP 'PO,nn	3 EX '(SP),HL	1 CALL 'PO,nn	3 PUSH 'HL	1 AND 'n	2 RST '20H	1
F RET 'P	1 POP 'AF	1 JP 'P,nn	3 DI	1 CALL 'P,nn	3 PUSH 'AF	1 OR 'n	2 RST '30H	1

INSTRUCTIONS DONT LE CODE EST SUR 1 OCTET (dernière partie)

	B	9	A	B	C	D	E	F	
0	EX 'AF, AF	1 ADD 'HL, BC	1 LD 'A, (BC)	1 DEC 'BC	1 INC 'C	1 DEC 'C	1 LD 'C, n	2 RRCA '	1
1	JR 'e	2 ADD 'HL, DE	1 LD 'A, (DE)	1 DEC 'DE	1 INC 'E	1 DEC 'E	1 LD 'E, n	2 RRA '	1
2	JR 'Z, e	2 ADD 'HL, HL	1 LD 'HL, (nn)	3 DEC 'HL	1 INC 'L	1 DEC 'L	1 LD 'L, n	2 CPL '	1
3	JR 'C, e	2 ADD 'HL, SP	1 LD 'A, (nn)	3 DEC 'SP	1 INC 'A	1 DEC 'A	1 LD 'A, n	2 CCF '	1
4	LD 'C, B	1 LD 'C, C	1 LD 'C, D	1 LD 'C, E	1 LD 'C, H	1 LD 'C, L	1 LD 'C, (HL)	1 LD 'C, A	1
5	LD 'E, B	1 LD 'E, C	1 LD 'E, D	1 LD 'E, E	1 LD 'E, H	1 LD 'E, L	1 LD 'E, (HL)	1 LD 'E, A	1
6	LD 'L, B	1 LD 'L, C	1 LD 'L, D	1 LD 'L, E	1 LD 'L, H	1 LD 'L, L	1 LD 'L, (HL)	1 LD 'L, A	1
7	LD 'A, B	1 LD 'A, C	1 LD 'A, D	1 LD 'A, E	1 LD 'A, H	1 LD 'A, L	1 LD 'A, (HL)	1 LD 'A, A	1
8	ADC 'A, B	1 ADC 'A, C	1 ADC 'A, D	1 ADC 'A, E	1 ADC 'A, H	1 ADC 'A, L	1 ADC 'A, (HL)	1 ADC 'A, A	1
9	SBC 'A, B	1 SBC 'A, C	1 SBC 'A, D	1 SBC 'A, E	1 SBC 'A, H	1 SBC 'A, L	1 SBC 'A, (HL)	1 SBC 'A, A	1
A	XOR 'B	1 XOR 'C	1 XOR 'D	1 XOR 'E	1 XOR 'H	1 XOR 'L	1 XOR '(HL)	1 XOR 'A	1
B	CP 'B	1 CP 'C	1 CP 'D	1 CP 'E	1 CP 'H	1 CP 'L	1 CP '(HL)	1 CP 'A	1
C	RET 'Z	1 RET '	1 JP 'Z, nn	3 '.....	CALL 'Z, nn	3 CALL 'nn	3 ADC 'A, n	2 RST '0BH	1
D	RET 'C	1 EXX '	1 JP 'C, nn	3 IN 'A, (n)	2 CALL 'C, nn	3 '.....	SBC 'A, n	2 RST '1BH	1
E	RET 'PE	1 JP '(HL)	1 JP 'PE, nn	3 EX 'DE, HL	1 CALL 'PE, nn	3 '.....	XOR 'n	2 RST '2BH	1
F	RET 'M	1 LD 'SP, HL	1 JP 'M, nn	3 EI '	1 CALL 'M, nn	3 '.....	CP 'n	2 RST '3BH	1

REMARQUE: les codes CB, DD, ED, et FD sont réservés pour les

instructions ayant un code instruction de 2 ou 3 octets.

INSTRUCTIONS DONT LE CODE EST SUR 2 OCTETS (1ère partie)

Instructions dont le 1er octet est CB;

Le 2ème octet donne les instructions suivant le tableau.

	0	1	2	3	4	5	6	7
0	RLC 'B	RLC 'C	RLC 'D	RLC 'E	RLC 'H	RLC 'L	RLC '(HL)	RLC 'A
1	RL 'B	RL 'C	RL 'D	RL 'E	RL 'H	RL 'L	RL '(HL)	RL 'A
2	SLA 'B	SLA 'C	SLA 'D	SLA 'E	SLA 'H	SLA 'L	SLA '(HL)	SLA 'A
3								
4	BIT '0,B	BIT '0,C	BIT '0,D	BIT '0,E	BIT '0,H	BIT '0,L	BIT '0,(HL)	BIT '0,A
5	BIT '2,B	BIT '2,C	BIT '2,D	BIT '2,E	BIT '2,H	BIT '2,L	BIT '2,(HL)	BIT '2,A
6	BIT '4,B	BIT '4,C	BIT '4,D	BIT '4,E	BIT '4,H	BIT '4,L	BIT '4,(HL)	BIT '4,A
7	BIT '6,B	BIT '6,C	BIT '6,D	BIT '6,E	BIT '6,H	BIT '6,L	BIT '6,(HL)	BIT '6,A
8	RES '0,B	RES '0,C	RES '0,D	RES '0,E	RES '0,H	RES '0,L	RES '0,(HL)	RES '0,A
9	RES '2,B	RES '2,C	RES '2,D	RES '2,E	RES '2,H	RES '2,L	RES '2,(HL)	RES '2,A
A	RES '4,B	RES '4,C	RES '4,D	RES '4,E	RES '4,H	RES '4,L	RES '4,(HL)	RES '4,A
B	RES '6,B	RES '6,C	RES '6,D	RES '6,E	RES '6,H	RES '6,L	RES '6,(HL)	RES '6,A
C	SET '0,B	SET '0,C	SET '0,D	SET '0,E	SET '0,H	SET '0,L	SET '0,(HL)	SET '0,A
D	SET '2,B	SET '2,C	SET '2,D	SET '2,E	SET '2,H	SET '2,L	SET '2,(HL)	SET '2,A
E	SET '4,B	SET '4,C	SET '4,D	SET '4,E	SET '4,H	SET '4,L	SET '4,(HL)	SET '4,A
F	SET '6,B	SET '6,C	SET '6,D	SET '6,E	SET '6,H	SET '6,L	SET '6,(HL)	SET '6,A

INSTRUCTIONS DONT LE CODE EST SUR 2 OCTETS (2ème partie)

Instructions dont le 1er octet est CB;

Le 2ème octet donne l'instruction suivant le tableau.

	B	9	A	B	C	D	E	F
0	RRC B	RRC C	RRC D	RRC E	RRC H	RRC L	RRC (HL)	RRC A
1	RR B	RR C	RR D	RR E	RR H	RR L	RR (HL)	RR A
2	SRA B	SRA C	SRA D	SRA E	SRA H	SRA L	SRA (HL)	SRA A
3	SRL B	SRL C	SRL D	SRL E	SRL H	SRL L	SRL (HL)	SRL A
4	BIT 1, B	BIT 1, C	BIT 1, D	BIT 1, E	BIT 1, H	BIT 1, L	BIT 1, (HL)	BIT 1, A
5	BIT 3, B	BIT 3, C	BIT 3, D	BIT 3, E	BIT 3, H	BIT 3, L	BIT 3, (HL)	BIT 3, A
6	BIT 5, B	BIT 5, C	BIT 5, D	BIT 5, E	BIT 5, H	BIT 5, L	BIT 5, (HL)	BIT 5, A
7	BIT 7, B	BIT 7, C	BIT 7, D	BIT 7, E	BIT 7, H	BIT 7, L	BIT 7, (HL)	BIT 7, A
8	RES 1, B	RES 1, C	RES 1, D	RES 1, E	RES 1, H	RES 1, L	RES 1, (HL)	RES 1, A
9	RES 3, B	RES 3, C	RES 3, D	RES 3, E	RES 3, H	RES 3, L	RES 3, (HL)	RES 3, A
A	RES 5, B	RES 5, C	RES 5, D	RES 5, E	RES 5, H	RES 5, L	RES 5, (HL)	RES 5, A
B	RES 7, B	RES 7, C	RES 7, D	RES 7, E	RES 7, H	RES 7, L	RES 7, (HL)	RES 7, A
C	SET 1, B	SET 1, C	SET 1, D	SET 1, E	SET 1, H	SET 1, L	SET 1, (HL)	SET 1, A
D	SET 3, B	SET 3, C	SET 3, D	SET 3, E	SET 3, H	SET 3, L	SET 3, (HL)	SET 3, A
E	SET 5, B	SET 5, C	SET 5, D	SET 5, E	SET 5, H	SET 5, L	SET 5, (HL)	SET 5, A
F	SET 7, B	SET 7, C	SET 7, D	SET 7, E	SET 7, H	SET 7, L	SET 7, (HL)	SET 7, A

INSTRUCTIONS DONT LE CODE EST SUR 2 OCTETS (3ème partie)

Instructions dont le 1er octet est DD;

Le 2ème octet donne l'instructions suivant le tableau.

	0	1	2	3	4	5	6
0							
1							
2		LD IX,nn	4 LD (nn),IX	4 INC IX	2		
3					INC (IX+d)	3 DEC (IX+d)	3 LD (IX+d),n
4							LD B, (IX+d)
5							LD D, (IX+d)
6							LD H, (IX+d)
7	LD (IX+d),B	3 LD (IX+d),C	3 LD (IX+d),D	3 LD (IX+d),E	3 LD (IX+d),H	3 LD (IX+d),L	
8							ADD A, (IX+d)
9							SUB (IX+d)
A							AND (IX+d)
B							OR (IX+d)
C							
D							
E		POP IX	2	EX (SP),IX	2	PUSH IX	2
F							

INSTRUCTIONS DONT LE CODE EST SUR 2 OCTETS (4ème partie)

Instructions dont le 1er octet est DD;
Le 2ème octet donne l'instruction suivant le tableau.

	7	9	A	E				
0		ADD IX, BC	2					
1		ADD IX, DE	2	LD IX, (nn)	4			
2		ADD IX, SP	2					
3								
4				LD C, (IX+d)	3			
5				LD E, (IX+d)	3			
6				LD L, (IX+d)	3			
7	LD (IX+d), A			LD A, (IX+d)	3			
8				ADC A, (IX+d)	3			
9				SBC A, (IX+d)	3			
A				XOR (IX+d)	3			
B				CF (IX+d)	3			
C								
D								
E								
F		JP (IX)	2					

INSTRUCTIONS DONT LE CODE EST SUR 2 OCTETS (5ème partie)

Instructions dont le 1er octet est FD;

Le 2ème octet donne l'instruction suivant le tableau.

	0	1	2	3	4	5	6
0							
1							
2		LD IY,nn	4 LD (nn),IY	4 INC IY	2		
3					INC (IY+d)	3 DEC (IY+d)	3 LD (IY+d),n
4							LD B, (IY+d)
5							LD D, (IY+d)
6							LD H, (IY+d)
7	LD (IY+d),B	3 LD (IY+d),C	3 LD (IY+d),D	3 LD (IY+d),E	3 LD (IY+d),H	3 LD (IY+d),L	
8							ADD A, (IY+d)
9							SUB (IY+d)
A							AND (IY+d)
B							OR (IY+d)
C							
D							
E		FOP IY	2	EX (SP),IY	2	PUSH IY	2
F							

INSTRUCTIONS DONT LE CODE EST SUR 2 OCTETS (6ème partie)

Instructions dont le 1er octet est FD;

Le 2ème octet donne l'instruction suivant le tableau.

	7	9	A	E				
0		ADD IY, BC	2					
1		ADD IY, DE	2	LD IY, (nn)	4			
2		ADD IY, SP	2					
3								
4				LD C, (IY+d)	3			
5				LD E, (IY+d)	3			
6				LD L, (IY+d)	3			
7	LD (IY+d), A			LD A, (IY+d)	3			
8				ADC A, (IY+d)	3			
9				SBC A, (IY+d)	3			
A				XOR (IY+d)	3			
B				CP (IY+d)	3			
C								
D								
E								
F		JP (IY)	2					

INSTRUCTIONS DONT LE CODE EST SUR 2 OCTETS (7ème partie)

Instructions dont le 1er octet est ED;

Le 2ème octet donne l'instruction suivant le tableau.

	0	1	2	3	4	5	6	7
0								
1								
2								
3								
4	IN B, (C)	OUT (C), B	SBC HL, BC	LD (nn), BC	NEG	RETN	IM 0	
5	IN D, (C)	OUT (C), D	SBC HL, DE	LD (nn), DE			IM 1	
6	IN H, (C)	OUT (C), H	SBC HL, HL					RRD
7			SBC HL, SP	LD (nn), SP				
8								
9								
A	LDI	CPI	INI	OUTI				
B	LDIR	CPIR	INIR	OTIR				
C								
D								
E								
F								

INSTRUCTIONS DONT LE CODE EST SUR 2 OCTETS (dernière partie)

Instructions dont le 1er octet est ED;

Le 2ème octet donne l'instruction suivant le tableau.

	B	9	A	B	C	D	E	F
0								
1								
2								
3								
4	IN 2 (C, (C)	OUT 2 (C), C	ADC 2 HL, BC	LD 4 BC, (nn)		RETI 2		LD 2 R, A
5	IN 2 (E, (C)	OUT 2 (C), E	ADC 2 HL, DE	LD 4 DE, (nn)			IM 2	LD 2 A, R
6	IN 2 L, (C)	OUT 2 (C), L	ADC 2 HL, HL					RLD 2
7	IN 2 A, (C)	OUT 2 (C), A	ADC 2 HL, SP	LD 4 SP, (nn)				
8				OTDR 2				
9								
A	LDD 2	CPD 2	IND 2	OUTD 2				
B	LDDR 2	CPDR 2	INDR 2					
C								
D								
E								
F								

INSTRUCTIONS DONT LE CODE EST SUR 3 OCTETS.

Instructions commençant
par DDCB.Instructions commençant
par FDCB.

Le 4ème octet donne l'instruction voulue.

6	E
0!RLC	4!RRC 4!
! (IX+d)	! (IX+d) !
1!RL	4!RR 4!
! (IX+d)	! (IX+d) !
2!SLA	4!SRA 4!
! (IX+d)	! (IX+d) !
3!SRL	4!
! (IX+d)	!
4!BIT	4!BIT 4!
! 0, (IX+d)	! 1, (IX+d) !
5!BIT	4!BIT 4!
! 2, (IX+d)	! 3, (IX+d) !
6!BIT	4!BIT 4!
! 4, (IX+d)	! 5, (IX+d) !
7!BIT	4!BIT 4!
! 6, (IX+d)	! 7, (IX+d) !
8!RES	4!RES 4!
! 0, (IX+d)	! 1, (IX+d) !
9!RES	4!RES 4!
! 2, (IX+d)	! 3, (IX+d) !
A!RES	4!RES 4!
! 4, (IX+d)	! 5, (IX+d) !
B!RES	4!RES 4!
! 6, (IX+d)	! 7, (IX+d) !
C!SET	4!SET 4!
! 0, (IX+d)	! 1, (IX+d) !
D!SET	4!SET 4!
! 2, (IX+d)	! 3, (IX+d) !
E!SET	4!SET 4!
! 4, (IX+d)	! 5, (IX+d) !
F!SET	4!SET 4!
! 6, (IX+d)	! 7, (IX+d) !

=====

<-- 4ème octet -->

=====

6	E
0!RLC	4!RRC 4!
! (IY+d)	! (IY+d) !
1!RL	4!RR 4!
! (IY+d)	! (IY+d) !
2!SLA	4!SRA 4!
! (IY+d)	! (IY+d) !
3!SRL	4!
! (IY+d)	!
4!BIT	4!BIT 4!
! 0, (IY+d)	! 1, (IY+d) !
5!BIT	4!BIT 4!
! 2, (IY+d)	! 3, (IY+d) !
6!BIT	4!BIT 4!
! 4, (IY+d)	! 5, (IY+d) !
7!BIT	4!BIT 4!
! 6, (IY+d)	! 7, (IY+d) !
8!RES	4!RES 4!
! 0, (IY+d)	! 1, (IY+d) !
9!RES	4!RES 4!
! 2, (IY+d)	! 3, (IY+d) !
A!RES	4!RES 4!
! 4, (IY+d)	! 5, (IY+d) !
B!RES	4!RES 4!
! 6, (IY+d)	! 7, (IY+d) !
C!SET	4!SET 4!
! 0, (IY+d)	! 1, (IY+d) !
D!SET	4!SET 4!
! 2, (IY+d)	! 3, (IY+d) !
E!SET	4!SET 4!
! 4, (IY+d)	! 5, (IY+d) !
F!SET	4!SET 4!
! 6, (IY+d)	! 7, (IY+d) !

ANNEXE 2 - PROGRAMMES DIVERS:

I - PRESENTATION:

Dans les pages qui vont suivre, il vous est présenté 4 programmes. Ces programmes, en BASIC, en ASSEMBLEUR, ou les deux à la fois, vous permettront de mettre en pratique tous les éléments qui ont été étudiés tout au long de cet ouvrage.

PREMIER PROGRAMME: (voir utilisation au chapitre 3)

Ce programme vous permettra d'effectuer la transformation d'un nombre décimal, entier ou flottant, positif ou négatif, en un nombre binaire (suite de 0 et de 1).

DEUXIEME PROGRAMME: (utilisé pour l'analyse des routines)

Ce programme vous permettra d'analyser vos routines particulières. Il peut s'avérer utile lors de la mise au point de programmes assembleur; le principe est d'initialiser les registres avec une valeur quelconque (ici &45), et de regarder, après exécution de la routine désirée (ici CALL &BC6E, en &7100), l'état de tous les registres et indicateurs (reg F) (ici en &8000). LA routine à analyser doit être mise à la place de l'instruction 35, dans le listing proposé. La sauvegarde des registres, après exécution, est faite à partir de l'adresse &8000, comme suit:

F anc / F nouv / IX / IY / C / B / E / D / L / H / A

&8000 —1 —2 —3 —4 —5 —6 —7 —8 —9 —A

Il faut être très vigilant quant à l'emploi de cette routine. En effet, les routines à analyser vous montreront l'état des registres, après l'exécution de la dernière instruction de votre routine spécifique; à vous de découper judicieusement votre programme assembleur pour une fine analyse...

TROISIEME PROGRAMME: (voir utilisation au chapitre 8)

Ce programme est une application directe de l'utilisation des routines cassettes; sa fonction n'est pas de "pirater" des logiciels du commerce, mais de sauvegarder ses propres logiciels (personnels ou achetés), afin de se prémunir des destructions accidentelles de ceux-ci.

Outre la fonction de copie de fichier (programmes ou données), il permet la constitution de listes de catalogues. Il est possible de lire des fichiers ayant un nombre de blocs maximum (24); mais cette opération doit se faire en 2 temps, car ce logiciel ne peut lire que 12 blocs en une fois; il suffit alors de sauver les 12 premiers blocs, puis de procéder au chargement des derniers blocs à copier.

Ce logiciel permet de changer le nom ou le type d'un fichier. Il permet également de changer un code particulier (1 octet); par exemple, de changer toutes les * d'un fichier en \$...

QUATRIEME PROGRAMME: (voir utilisation au chapitre 3)

Ce programme DUMP, offre à l'utilisateur de nombreuses fonctions. Grâce au menu vous pouvez voir qu'il est possible d'effectuer des dump en RAM comme en ROM; vous pouvez également effectuer des transferts de blocs d'octets pris en RAM comme en ROM.

Vous pouvez aussi créer de petites routines assembleur, en entrant les codes HEXA, et sauver vos fichiers pour les relire plus tard. Vous pouvez par ailleurs travailler indifféremment sur cassette ou sur disquette, sur imprimante ou sur écran.

En ce qui concerne les dumps sur imprimante, il vous est possible de fixer 2 tabulations (menu B), afin de concevoir des cadrages plus esthétiques.

II - TRANSFORMATION DU DECIMAL EN BINAIRE:

```

10 REM =====
20 REM =          DECIMAL-BINAIRE          =
30 REM = ROY DIDIER - ORLEANS / 85 =
40 REM =====
45 ON BREAK GOSUB 600
50 CLS:LOCATE 12,4:PRINT"  NOMBRE DECIMAL"
60 LOCATE 12,5:PRINT"ENTIER OU FLOTTANT"
70 WINDOW £1,12,29,7,9:PAPER £1,1:PEN £1,3:CLS £1
80 LOCATE 12,12:PRINT"TRADUCTION BINAIRE"
90 WINDOW £2,4,37,14,16:PAPER £2,2:CLS £2
100 LOCATE 18,18:PRINT"CHOIX ?"
110 WINDOW £3,10,31,20,22:PAPER £3,3:CLS £3:PRINT£3:PRINT£3,"1 - SU
ITE / 2 - ARRET"
120 PRINT£1:INPUT £1,D$:D=VAL(D$)
130 IF D=0 THEN B$="0":GOTO 500
140 IF D>0 THEN B$="+" ELSE B$="-":D=ABS(D)
150 P=-1:GOSUB 500:IF D<1 THEN B$=B$+ "." :GOTO 180
160 IF D>0 THEN P=P+1:GOSUB 500:GOTO 160
170 IF D<0 THEN P=P-1:GOSUB 500:GOTO 170
180 IF D<0 THEN B$=B$+"0":GOTO 200
190 D=D-Q:B$=B$+"1":IF LEN(B$)>33 THEN 300
200 P=P-1:IF P=-1 THEN B$=B$+ "."
210 GOSUB 500:IF LEN(B$)<34 THEN 180
300 PRINT£2:PRINT£2,B$
310 A$=INKEY$:IF A$="" THEN 310
320 IF A$<>"1" THEN MODE 1:END
330 CLS £1:CLS £2:GOTO 120
500 Q=INT(2^ABS(P)):IF P<0 THEN Q=1/Q
510 RETURN
600 MODE 1:END

```

III - PROGRAMME D'ANALYSE DES REGISTRES:

1		
2		ORG 7000H
3		LOAD \$
4	7000 210000	LD HL,0
5	7003 08	EX AF,AF
6	7004 39	ADD HL,SP
7	7005 08	EX AF,AF
8	7006 F5	PUSH AF
9	7007 2B	DEC HL
10	7008 2B	DEC HL
11	7009 7E	LD A,(HL)
12	700A 320080	LD (8000H),A
13	700D F1	POP AF
14	700E 3E45	LD A,45H
15	7010 214545	LD HL,4545H
16	7013 114545	LD DE,4545H
17	7016 014545	LD BC,4545H
18	7019 DD214500	LD IX,45H
19	701D FD214500	LD IY,45H
20	7021 CD0071	CALL 7100H
21	7024 DD220280	LD (8002H),IX
22	7028 FD220380	LD (8003H),IY
23	702C 320A80	LD (800AH),A
24	702F ED430480	LD (8004H),BC
25	7033 ED530680	LD (8006H),DE
26	7037 220880	LD (8008H),HL
27	703A 210000	LD HL,0
28	703D 08	EX AF,AF
29	703E 39	ADD HL,SP
30	703F 08	EX AF,AF
31	7040 F5	PUSH AF
32	7041 2B	DEC HL
33	7042 2B	DEC HL
34	7043 7E	LD A,(HL)
35	7044 320180	LD (8001H),A
36	7047 F1	POP AF
37	7048 C9	RET
38		ORG 7100H
39		LOAD \$
40	7100 CD6EBC	CALL 0BC6EH
41	7103 C9	RET
42		END

IV - SAUVEGARDE DE LOGICIELS ET FICHIERS PERSONNELS:

```

10 REM =====
20 REM =      SUPER  SAUVEGARDE      =
30 REM = ROY DIDIER - ORLEANS / 85 =
40 REM =====
50 MEMORY 8500:RESTORE:CALL &BC71
60 ON BREAK GOSUB 1300
80 MODE 1:BORDER 13:PEN £2,3:GOSUB 680
90 WINDOW £1,6,35,2,4:PAPER £1,1:PEN £1,0:CLS £1
100 WINDOW £2,2,39,6,18:PAPER £2,2:CLS £2
110 WINDOW £3,2,39,20,24:PAPER £3,3:PEN £3,1:CLS £3
120 PRINT £1:PRINT £1,"      Mettre la K7 a copier":PRINT £1:PRINT £1
    ," Presser PLAY puis une touche"
130 GOSUB 1610
140 PRINT£3:PRINT£3:PRINT£3,"  SAUVEGARDE DE LOGICIELS PERSONNELS"
185 a$=INKEY$:IF a$="" THEN 185
190 CLS £1:CLS £3
200 GOSUB 920:IF kat=1 GOTO 220
210 PRINT £3:PRINT £3:PRINT £3,"      Chargement logiciel en cours"
220 g=35:f=9100:q=f:CALL &BC6E
230 CALL 9000:GOSUB 740
240 IF kat=1 THEN CALL &BC71:GOTO 1000
250 r=PEEK(f+19):s=PEEK(f+20)
260 POKE 9015,r:POKE 9016,s:CALL 9012
270 IF PEEK (f+17)<>0 THEN CLS £3:GOTO 350
290 IF g=145 THEN GOTO 310
300 f=f+&A00:g=g+10:q=f:POKE 9007,g:POKE 9019,g:GOTO 230
310 CLS £3:PRINT £3:PRINT £3,"      Memoire pleine ..."
320 PRINT £3,"  Sauvez le chargement qui a ete fait"
330 PRINT £3,"      Ensuite retour au chargement"
340 PRINT £3,"      pour completer le logiciel"
350 CALL &BC71:CLS £1:CLS £2:PRINT £1
360 PRINT£1:PRINT £1,"      Mettre un blanc et presser"
370 PRINT£1,TAB(12)"REC/PLAY"
380 PRINT £2:PRINT £2,"  Choisissez la vitesse / protection":PRINT£2
400 PRINT £2:PRINT £2,"  Touche 1 (lent/pas de protection)"
410 PRINT £2:PRINT £2,"  Touche 2 (rapide/pas de protection)"
420 PRINT £2:PRINT £2,"  Touche 3 (lent/protege)"
430 PRINT £2:PRINT £2,"  Touche 4 (rapide/protege)"
440 PRINT £2:PRINT £2,TAB(14)"choix ? -->"
450 a$=INKEY$:IF a$="" THEN 450
460 IF a$<>"1" AND a$<>"2" AND a$<>"3" AND a$<>"4" GOTO 450
470 CLS £1:CLS £2:GOSUB 1600

```

```

480 IF a$="1" THEN SPEED WRITE 0:GOSUB 650:GOTO 520
490 IF a$="2" THEN SPEED WRITE 1:GOSUB 650:GOTO 520
500 IF a$="3" THEN SPEED WRITE 0:GOSUB 620:GOTO 520
510 IF a$="4" THEN SPEED WRITE 1:GOSUB 620:GOTO 520
520 GOSUB 1040:GOSUB 1310:GOSUB 1460
530 CLS £3:PRINT £3:PRINT £3:PRINT £3,"          Sauvegarde logiciel en
cours"
540 CALL &BC6E:FOR Z=1 TO 2000:NEXT:CALL &BC71
550 q=9100:m=35:CALL &BC6E
560 GOSUB 740:CALL 9024:r=PEEK(q+19):s=PEEK(q+20)
570 POKE 9039,r:POKE 9040,s:CALL 9036
580 IF PEEK(q+17)<>0 THEN GOTO 1250
590 IF m=q THEN GOTO 1250
600 q=q+&A00:m=m+10:POKE 9031,m:POKE 9043,m
610 FOR X=1 TO 1250:NEXT:GOTO 560
620 REM === PROTECTION ===
625 J=9100
630 u=j+18:IF PEEK(u)=0 THEN POKE u,1:IF PEEK(u)=2 THEN POKE u,3
640 IF J=f THEN RETURN ELSE J=J+&A00:GOTO 630
650 REM === PAS DE PROTECTION ===
655 J=9100
660 u=j+18:IF PEEK(u)=1 THEN POKE u,0:IF PEEK(u)=3 THEN POKE u,2
670 IF J=f THEN RETURN ELSE J=J+&A00:GOTO 660
680 REM === CHARG CODE MACHINE ===
685 FOR y=9000 TO 9047:READ D$:D$="&"&D$:D=VAL(D$):POKE y,D:NEXT
690 DATA 3E,2C,11,40,0,21,8C,23,CD,A1,BC,C9
700 DATA 3E,16,11,0,0,21,CD,23,CD,A1,BC,C9
710 DATA 3E,2C,11,40,0,21,8C,23,CD,9E,BC,C9
720 DATA 3E,16,11,0,0,21,CD,23,CD,9E,BC,C9
730 RETURN
740 REM === AFFICHAGE CATALOGUE ===
745 IF kat=0 THEN w=2
750 CLS £1:PRINT £1:PRINT £1,SPACE$(8)"INFORMATIONS"
760 CLS £2:PRINT £w:PRINT £w," NOM DE FICHIER..... ":IF PEEK(q)=
0 THEN PRINT £w,"Pas de nom";ELSE FOR e=q TO (q+15):IF PEEK(q)<>0
THEN PRINT £w,CHR$(PEEK(e));:NEXT
770 PRINT £2:PRINT £w:PRINT £w," BLOC NUMERO.....":PEEK(q+16);
:IF PEEK(q+17)<>0 THEN PRINT £w,"Dernier bloc"
780 PRINT £w:LOCATE £2,1,6:PRINT £w," TYPE DE FICHIER.... ";
790 t=PEEK(q+18)
800 IF t=0 THEN PRINT £w,"Basic"
810 IF t=1 THEN PRINT £w,"Basic protege"
820 IF t=2 THEN PRINT £w,"Binaire"
830 IF t=3 THEN PRINT £w,"Binaire protege"
840 IF t=4 OR t=5 THEN PRINT £w,"Image ecran"
850 IF t=6 THEN PRINT £w,"Ascii"

```

```

860 PRINT £w:PRINT £w," LONGUEUR DONNEES....";PEEK (q+19)+PEEK(q+20)
)*%100
870 PRINT £2:PRINT £w," ADRESSE DEBUT.....";PEEK (q+21)+PEEK(q+22)
)*%100
880 PRINT £2:PRINT £w," LONGUEUR FICHIER....";PEEK (q+24)+PEEK(q+25)
)*%100
890 PRINT £2:PRINT £w," ADRESSE D'EXECUTION.":IF PEEK (q+26)+PEEK(
q+27)*%100 = 0 THEN PRINT £w," (Non donnee)":ELSE PRINT £w,PEEK (q+
26)+PEEK(q+27)*%100
900 W=2:RETURN
920 REM === CHOIX CATALOGUE ===
925 FOR i=1 TO 1000:NEXT:CLS £1:PRINT£1
930 PRINT£1," voulez-vous lire seulement"
940 PRINT£1," le catalogue ? ";
950 a$=INKEY$:IF a$="" GOTO 950
960 PRINT£1,a$
970 IF a$="O" OR a$="o" THEN kat=1:GOSUB 1200
980 IF kat=1 THEN PRINT£3:PRINT£3:PRINT£3," Lecture cat. en
cours"
990 RETURN
1000 REM === RETOUR DEBUT ===
1005 CLS £3:CLS £1:PRINT£1," Appuyez sur une touche"
1010 PRINT£1," pour continuer"
1020 a$=INKEY$:IF a$="" GOTO 1020
1030 RUN
1040 REM === MODIF DU NOM ===
1045 j=9100
1050 PRINT£1:PRINT£1," Voulez-vous changer"
1060 PRINT£1," le nom du logiciel ? ";
1070 a$=INKEY$:IF a$="" GOTO 1070
1080 PRINT£1,a$
1090 IF a$<>"O" AND a$<>"o" THEN RETURN
1100 CLS £1
1110 PRINT£1:INPUT £1," NOM DES. : ";n$
1120 l=LEN(n$)
1130 IF l>16 GOTO 1100
1140 FOR k=0 TO 15:POKE (j+k),0:NEXT
1150 FOR k=0 TO l-1
1160 x$=MID$(n$,k+1,1):x=ASC(x$):POKE (j+k),x
1170 NEXT
1180 IF j=f THEN RETURN
1190 j=j+%A00:GOTO 1140

```

```

1200 REM === CHOIX IMPRIMANTE ===
1205 PRINT#1:PRINT#1,"          Imprimante (o/n) ? ";
1210 a$=INKEY$:IF a$="" GOTO 1210
1220 PRINT#1,a$
1230 IF a$="0" OR a$="o" THEN w=8 ELSE w=2
1240 RETURN
1250 REM === AUTRE SAUVEGARDE ===
1255 CALL %BC71:CLS #3:PRINT#3
1260 PRINT#3,"          Autre sauvegarde (o/n) ? ";
1270 a$=INKEY$:IF a$="" GOTO 1270
1280 PRINT#3,a$:IF a$="0" OR a$="o" THEN CLS #3:GOTO 350
1290 CALL %BB00:RUN
1300 REM === FIN PROGRAMME ===
1305 MEMORY %9000:CLS: BORDER 0,0:END
1310 REM === MODIF DU TYPE ===
1315 j=9100
1320 PRINT#1:PRINT#1,"          voulez-vous changer"
1330 PRINT#1,"          le type du fichier ?";
1340 a$=INKEY$:IF a$="" GOTO 1340
1350 PRINT#1,a$
1360 IF a$<>"o" AND a$<>"0" THEN RETURN
1370 CLS #2:PRINT#2:PRINT#2
1380 PRINT#2,TAB(15)"0 - basic"
1390 PRINT#2,TAB(15)"2 - binaire"
1400 PRINT#2,TAB(15)"6 - ASCII"
1410 PRINT#2:INPUT #2,"          choix --> ";t
1420 IF t<>0 AND t<>2 AND t<>6 THEN 1370
1430 POKE (j+18),t
1440 IF J=f THEN CLS #2:RETURN
1450 j=j+%A00:GOTO 1430
1460 REM === MODIF D'UN CODE ===
1465 j=9100
1470 PRINT#1:PRINT#1,"          voulez-vous changer"
1480 PRINT#1,"          un code du fichier ?";
1490 a$=INKEY$:IF a$="" GOTO 1490
1500 PRINT#1,a$
1510 IF a$<>"o" AND a$<>"0" THEN RETURN
1520 CLS #2:PRINT#2
1530 INPUT #2,"code a changer --> ";ac
1540 INPUT #2,"nouveau code --> ";nc
1550 FOR i=j+65 TO j+2500
1560 IF PEEK(i)=ac THEN POKE i,nc
1570 NEXT
1580 IF j=f THEN RETURN
1590 j=j+%A00:GOTO 1550
1600 REM === PRESENTATION ===
1605 LOCATE #2,13,4:PRINT#2,"SUPER COPIEUR"
1610 LOCATE #2,6,7:PRINT #2,"          *** ROY Didier ***"
1620 LOCATE #2,6,10:PRINT #2,"Fait a Orleans en mars 1985"
1630 RETURN

```

```

10 REM =====
20 REM = Programme SUPER-DUMP =
30 REM = Roy Didier - Orleans / 85 =
40 REM =====
45 MEMORY &3FFF
47 ON BREAK GOSUB 2100
50 GOSUB 300:GOSUB 1200
70 GOSUB 1000
80 c$=INKEY$:IF c$="" THEN 80
85 IF c$="a" OR c$="A" THEN GOSUB 2200:GOTO 70
87 IF c$="b" OR c$="B" THEN GOSUB 2500:GOTO 70
88 IF c$="c" OR c$="C" THEN GOSUB 2700:GOTO 70
90 c=VAL(c$):IF c<1 OR c>9 THEN 80
100 ON c GOTO 110,130,150,170,190,210,220,230,240
110 tb=0:GOSUB 400:GOTO 70
130 tb=1:GOSUB 400:GOTO 70
150 rr=0:GOSUB 1500:GOSUB 600:CLS #3
160 GOTO 70
170 rr=1:GOSUB 1600:GOSUB 600:CLS #3
180 GOTO 70
190 IF pp=2 THEN pp=0 ELSE pp=2
195 GOSUB 1700
200 GOTO 80
210 GOSUB 1300:GOTO 70
220 GOSUB 1800:GOTO 70
230 GOSUB 1900:GOTO 70
240 IF kd=0 THEN kd=1 ELSE kd=0
245 GOSUB 2000:GOTO 80
250 END
300 REM === changement codes machine ===
310 DATA &CD,0,&B9,&CD,6,&B9,1,0,0,&11,0,0,&21,0,0,&ED,&B0,&CD,3,&B9,&CD,9,&B9,&
C9
320 DATA &CD,0,&B9,&CD,6,&B9,&3A,0,0,&32,&30,&90,&CD,3,&B9,&CD,9,&B9,&C9
330 FOR I=&9000 TO &902A:READ C:POKE I,C:NEXT
340 RETURN
400 REM === transfert de blocs ===
410 CLS #2:PRINT#2:PRINT#2
420 INPUT #2," Adresse debut source --> ";ds:PRINT#2
430 INPUT #2," Adresse fin source --> ";fs:PRINT#2
440 INPUT #2," Adresse destination --> ";ad:PRINT#2:nb=fs-ds+1
445 IF ds>fs THEN 410
447 IF ad<&4000 AND ad>&9000-nb THEN 410
450 ds$=HEX$(ds,4):d1$="&"+RIGHT$(ds$,2):d2$="&"+LEFT$(ds$,2)
460 ad$=HEX$(ad,4):a1$="&"+RIGHT$(ad$,2):a2$="&"+LEFT$(ad$,2)
470 nb$=HEX$(nb,4):n1$="&"+RIGHT$(nb$,2):n2$="&"+LEFT$(nb$,2)
480 d1=VAL(d1$):d2=VAL(d2$):POKE &900D,d1:POKE &900E,d2
490 a1=VAL(a1$):a2=VAL(a2$):POKE &900A,a1:POKE &900B,a2
500 n1=VAL(n1$):n2=VAL(n2$):POKE &9007,n1:POKE &9008,n2
510 IF tb=0 THEN CALL &9006 ELSE CALL &9000
520 RETURN
600 REM === dump memoire ===
610 CLS #2:PRINT#2:PRINT#2
620 INPUT #2," Debut dump memoire --> ";d:PRINT#2
630 INPUT #2," Fin dump memoire --> ";f:PRINT#2
640 IF d<0 THEN d=65536+d
650 IF f<0 THEN f=65536+f
660 IF pp=2 THEN CLS #2:LOCATE #3,2,5:PRINT#3,"A = anet - autre touche = stop/s
uite"
670 b$="":c$=""
680 FOR i=d TO f STEP ww
690 FOR j=i TO i+ww-1
695 IF j>f THEN b$=b$+",";c$=c$+",";GOTO 760
700 IF rr=1 THEN GOSUB 900:GOTO 720
710 o=PEEK(j):o$=HEX$(o)

```

```

720 IF LEN(c$)=1 THEN c$="0"+c$
730 b$=b$+c$+" "
740 IF c<32 OR c=&8A OR c=&8D OR c=&8E OR c=&8F OR c=&8B OR c=&89 OR 0=&8C THEN
  c=46
750 c$=c$+CHR$(c)
760 NEXT
770 i$=INKEY$ IF i$<>" " GOTO 780 ELSE GOTO 800
780 IF i$="H" OR i$="a" THEN i=i+1 GOTO 820
790 i$=INKEY$ IF i$<>" " GOTO 800 ELSE GOTO 790
800 IF i$="H" OR i$="a" THEN i=i+1 GOTO 820
805 IF PP=2 THEN T1$="":T2$=""
810 z$=T1$+HEX$(i)+": "+T2$+b$+"! "+c$:PRINT #PP,z$:b$="":c$=""
820 NEXT
822 IF PP=8 THEN PRINT #8
825 CLS #3:PRINT#3:PRINT#3:" TAPEZ UNE TOUCHE POUR CONTINUER:"
827 i$=INKEY$ IF i$="" THEN 827
830 RETURN
900 REM === lecture octet en rom ===
910 y$=HEX$(j,4):x$="&"+LEFT$(xy$,2):y$="&"+RIGHT$(xy$,2):x=VAL(x$):y=VAL(y$)
920 POKE &9020,x:POKE &901F,y:CALL &9018
930 c=PEEK(&9030):o$=HEX$(c)
940 RETURN
1000 REM === affichage du menu ===
1010 CLS #2:CLS #3
1020 PRINT#2,TAB(5)"1 - Transfert bloc RAM -> RAM"
1025 PRINT#2,TAB(5)"2 - Transfert bloc ROM -> RAM"
1030 PRINT#2,TAB(5)"3 - Dump memoire RAM"
1040 PRINT#2,TAB(5)"4 - Dump memoire ROM"
1050 PRINT#2,TAB(5)"5 - Bascule Ecran/Imprimante"
1060 PRINT#2,TAB(5)"6 - Entree de codes machines"
1070 PRINT#2,TAB(5)"7 - Lecture fichier code"
1080 PRINT#2,TAB(5)"8 - Ecriture fichier code"
1090 PRINT#2,TAB(5)"9 - Bascule K7 / Disquette"
1095 PRINT#2,TAB(5)"A - Modification d'un code"
1097 PRINT#2,TAB(5)"D - Tabulation imPression"
1098 PRINT#2,TAB(5)"C - Entree de codes ASCII"
1100 PRINT#2:PRINT#2,TAB(15)"CHOIX ?":PRINT#2
1110 PRINT#2,TAB(3)"MEMOIRE DISPONIBLE -> &4000 - &8fff"
1120 RETURN
1200 REM === initialisation ecran ===
1210 MODE 1 BORDER 0 PEN #2,3:PP=2:kd=0:ww=8
1220 WINDOW #1,1,40,1,5:PAPER #1,3
1230 WINDOW #2,1,40,2,21:PAPER #2,10:CLS #2
1235 WINDOW #3,1,40,22,26:CLS #3
1240 CLO #1:PRINT#1:PRINT#1,TAB(15)"SUPER-DUMP"
1250 PRINT#1,TAB(15)"ROY DIDIER"
1260 PRINT#1,TAB(15)"ORLEANS 85"
1265 GOSUB 1700 GOSUB 2000
1270 RETURN
1300 REM === Entree de codes machines ===
1310 CLS #2:PRINT#2:PRINT#3
1320 PRINT#3,TAB(5)"VALEUR A ENTRER EN HEXA"
1330 PRINT#3,TAB(5)"TAPEZ FFF POUR TERMINER"
1340 INPUT #2;"Adresse de depart: ";ad$:PRINT#2
1350 IF ad$="fff" OR ad$="FFF" THEN RETURN
1360 ad$="&"+ad$:ad=VAL(ad$)
1370 PRINT#2,HEX$(ad):" "
1380 INPUT #2;"Valeur --> ";v$
1390 IF v$="fff" OR v$="FFF" THEN RETURN
1400 v$="&"+v$:v=VAL(v$)
1410 IF v>255 GOTO 1370
1420 POKE ad,v:ad=ad+1
1430 GOTO 1370
1500 REM === message menu 3 ===

```



```

1510 CLS #3:PRINT#3:PRINT#3
1520 PRINT#3,TAB(12)"DUMP MEMOIRE RAM"
1530 RETURN
1600 REM === message menu 4 ===
1610 CLS #3:PRINT#3:PRINT#3
1620 PRINT#3,TAB(12)"DUMP MEMOIRE ROM"
1630 RETURN
1700 REM === message menu 5 ===
1710 IF PP=2 THEN A$="E" ELSE A$="I"
1720 LOCATE #1,35,2:PRINT #1,"<"A$">"
1730 RETURN
1800 REM === lecture fichier code ===
1810 CLS #2:PRINT#2:PRINT#2:PRINT#2
1820 INPUT #2,"      Nom du fichier --> " ;n$
1830 PRINT#3:PRINT#3
1840 PRINT#3,TAB(8)"LECTURE DU FICHIER CODE"
1842 CLS #1:LOCATE 1,1:PRINT
1845 LOAD n$
1847 GOSUB 1240
1850 RETURN
1900 REM === ecriture du fichier code ===
1910 CLS #2:PRINT#2:PRINT#2:PRINT#2
1920 INPUT #2,"      Nom du fichier --> " ;n$
1930 INPUT #2,"      Debut du fichier --> " ;d
1940 INPUT #2,"      Fin du fichier --> " ;f
1945 IF d<0 THEN d=65536+d
1947 IF f<0 THEN f=65536+f
1950 l=f-d+1
1952 PRINT#3:PRINT#3
1955 PRINT#3,TAB(8)"ECRITURE DU FICHIER CODE"
1957 CLS #1:LOCATE 1,1:PRINT
1960 SAVE n$,b,d,l
1965 GOSUB 1240
1970 RETURN
2000 REM === cassette / disquette ===
2005 IF kd=0 THEN kd=0:TAPE
2007 IF kd=1 THEN kd=1:DISC
2010 IF kd=0 THEN a$="C" ELSE a$="D"
2020 LOCATE #1,3,2:PRINT #1,"<"a$">"
2030 RETURN
2100 REM === on break ===
2110 BORDER 0,0:PAPER 0:FEN 1:CLS
2120 END
2200 REM === modif code ===
2210 CLS #2:PRINT#2:PRINT#2
2220 INPUT #2,"      Code a changer --> " ;c:PRINT#2
2225 INPUT #2,"      Nouveau code --> " ;n:PRINT#2
2230 INPUT #2,"      Adresse debut --> " ;d:PRINT#2
2240 INPUT #2,"      Adresse fin --> " ;f:PRINT#2
2250 IF c>255 THEN 2210
2260 FOR i=d TO f
2270 IF PEEK(i)=c THEN POKE i,n
2280 NEXT
2290 RETURN
2300 REM === tabulation imp. ===
2310 CLS #2
2320 PRINT#2,"      = PRESENTATION DE L'IMPRESSION =" :PRINT#2
2330 PRINT#2,"tab-1/adr./tab-1/8 octets/ASCII"
2340 PRINT#2:PRINT#2,"Exemple:" :PRINT#2
2350 PRINT#2," 188: 20 44 55 4D 50 20 2A 2A ! DUMP **"
2360 PRINT#2,"! "
2370 PRINT#2,"! "
2380 PRINT#2,"! --> TAB-2"
2390 PRINT#2,"-----> TAB-1"

```

```

2600 PRINT#2: INPUT #2, "Nombre d'esPace Pour TAB-1 "; T1
2610 INPUT #2, "Nombre d'esPace Pour TAB-2 "; T2
2615 INPUT #2, "Nombre d'octets Par ligne: "; nnn
2620 T1$="" : T2$=""
2630 IF T1<0 THEN T1$=T1$+" " : T1=T1+1:GOTO 2630
2640 IF T2<0 THEN T2$=T2$+" " : T2=T2+1:GOTO 2640
2650 RETURN
2700 REM === entree de codes ASCII ===
2710 CLS #2:CLS #3:PRINT#3:PRINT#3,TAB(10)"ENTREE DE CODES ASCII":PRINT#2
2720 PRINT#2: INPUT #2, "Adresse debut: ";d$:d=VAL(d$)
2730 PRINT#2: INPUT #2, "Chaine ASCII: ";c$:l=LEN(c$)
2740 FOR i=1 TO l
2750 x$=MID$(c$,i,1):x=ASC(x$)
2760 POKE d,x:d=d+1
2770 NEXT
2780 RETURN

```

LEXIQUE DES MOTS OU EXPRESSIONS INFORMATIQUES

UTILISES:

ASCII: American Standard Code for Information Interchange; c'est à dire code standard pour la transmission de l'information. Bon nombre d'ordinateurs, et pratiquement tous les petits ordinateurs utilisent ce type de codage pour représenter les caractères alphabétiques, numériques, et spéciaux.

ASSEMBLEUR: Logiciel transformant un programme assembleur en un langage directement compréhensible par la machine, c'est à dire en binaire, mais dont la visualisation à l'écran se fait en hexadécimal.

AZERTY: Type de clavier dont la deuxième rangée à partir du haut, commence par les caractères A,Z,E,R,T, et Y.

BINAIRE: Système de numération utilisant les symboles 0 et 1.

BIT: Plus petite information en informatique (0 ou 1); équivalent au chiffre dans le système de numération décimal.

BLOC: Sur amstrad, ce terme signifie 2K de données (2048 octets).

BUFFER: Zone de réception de données, créée temporairement pour l'exécution d'un travail donné (ex: buffer de lecture de 2K)

DIGIT: équivalent du bit; on dit aussi digit binaire.

DESASSEMBLEUR: Logiciel traduisant un programme en langage machine, en langage assembleur.

DONNEE: Ce peut être un caractère (code ASCII) ou une valeur hexa représentant une commande, une variable, une adresse etc ...

DUMP: Logiciel de traduction des codes ASCII en caractères.

ENREGISTREMENT: Parties fixe et répétitive d'un fichier, sur les micro-ordinateurs; l'enregistrement est constitué de données élémentaires; un fichier peut être représenté par un seul enregistrement.

EXPOSANT: Partie d'un nombre binaire en virgule flottante indiquant par quelle puissance il faut multiplier la mantisse pour obtenir la valeur réelle du nombre considéré.

FENETRE: Rectangle définissant un espace donné sur l'écran; il peut y avoir plusieurs fenêtres dans un même espace écran; l'amstrad en autorise jusqu'à 8 simultanées.

FICHIER: Ensemble d'enregistrements ou de données, pouvant être un programme au sens le plus large; en général il s'agit des données d'un programme.

FLOTTANT: Nombre comportant une partie entière et une partie décimale.

HARDWARE: Partie matérielle d'un ordinateur (U.C, interface etc...)

HEXADECIMAL: Système de numération utilisant les symboles 0 à 9, et les symboles A à F.

HORLOGE: Élément d'un ordinateur utilisé pour la synchronisation de l'exécution des tâches (instructions, gestion organes).

INDICATEUR: Pour chaque type de micro-processeur, un registre regroupe les divers indicateurs utilisés par le programme assembleur, pour connaître le résultat de l'exécution des instructions précédentes; pour le Z80 c'est le registre F.

INTERFACE: Il s'agit de l'élément d'un ordinateur effectuant la transaction entre celui-ci et l'extérieur; une interface d'entrée-sortie permet la communication avec l'imprimante, l'unité de disquette etc ...

INTERPRETEUR BASIC: Logiciel traduisant les instructions basic en langage machine au moment de leurs apparitions successives. Dans un programme basic, une même instruction sera traduite autant de fois que le programmeur fera exécuter celle-ci. C'est pourquoi un basic est nettement moins performant qu'un programme assembleur, qui lui, est déjà en langage machine.

LOGICIEL: Equivalent de programme; toutefois on considère celui-ci comme un programme assez sophistiqué.

MANTISSE: Partie d'un nombre binaire en virgule flottante qui représente les chiffres significatifs du nombre considéré.

MATRICE DE CARACTERE: Représentation binaire sous la forme de 8 lignes d'octets (ex: huit FF représenteront un carré plein).

OCTET: Groupe de huit bits représentant un nombre, un caractère, un code de contrôle etc ...

PERIPHERIQUE: Unité interne ou externe d'un ordinateur, et qui communique avec celui-ci par l'intermédiaire d'une interface spécialisée; ce peut être une imprimante, un lecteur de cassettes etc...

PIXEL: Plus petit élément d'affichage représenté sur l'écran.

QUARTET: Moitié gauche ou droite d'un octet.

QWERTY: Type de clavier dont la deuxième rangée à partir du haut commence par les caractères Q,W,E,R,T, et Y.

RAM: Random Access Memory; c'est à dire une mémoire qui peut être lue ou écrite (mémoire utilisateur).

REGISTRE: Zones élémentaires réservées au micro-processeur pour utilisation par les programmes assembleurs; Le Z80 utilise un accumulateur (A), deux registres index (IX et IY), et trois paires de registres simples (BC, DE, et HL).

ROM: Read Only Memory; c'est à dire une mémoire qui ne peut être que lue (mémoire contenant le système).

ROUTINE: Petit programme, ou partie répétitive d'un programme plus important.

SCROLLING: Action de défilement des caractères sur l'écran; il peut être horizontal ou vertical.

SYSTEME D'EXPLOITATION: Logiciel supervisant le déroulement des diverses tâches d'un ordinateur (entrée-sortie, etc...).

UNITE CENTRALE: (U.C) Partie de l'ordinateur qui exécute les instructions.

MICRO APPLICATION MICRO APPLICATION

EAUTÉS NOUVEAUTÉS NOUV

LIVRE DU LECTEUR DE DISQUETTE AMSTRAD CPC (Tome 10)

Tout sur la programmation et la gestion des données avec le floppy DDI-1 et le 664 ! Utile au débutant comme au programmeur en langage machine. Contient le listing du DOS commenté, un utilitaire qui ajoute

les fichiers RELATIFS à l'AMDOS avec de nouvelles commandes BASIC, un MONITEUR disque et beaucoup d'autres programmes et astuces... Ce livre est indispensable à tous ceux qui utilisent un floppy ou un 664 AMSTRAD.

Ref. : ML127
Prix : 149 FF



LE NOUVEL ATARI ST

Ce livre décrit la superbe machine qu'est l'ATARI ST. Architecture, interfaces, operating system, le bios, GEM, LOGO, le processeur 68000, sont quelques-uns

des thèmes abordés. Ce livre doit être lu par tous ceux qui suivent de près le monde de la micro-informatique.

Ref. : ML125
Prix : 129 FF

LE NOUVEAU COMMODORE 128

Ce livre présente le nouveau Commodore 128. Vous y trouverez un aperçu complet des possibilités du successeur du célèbre "64" et une présentation détaillée des trois operating system. Le super nouveau

BASIC Commodore 7.0 est décrit ainsi que la configuration de la mémoire, la page zéro et le nouveau et rapide lecteur de disquette 1571. Pour tous les Commodoristes !

Ref. : ML130
Prix : 129 FF



MICRO APPLICATION MICRO APPLICATION

LES LIVRES AMSTRAD

TRUCS ET ASTUCES POUR L'AMSTRAD CPC (Tome 1)

C'est le livre que tout utilisateur d'un CPC doit posséder. De nombreux domaines sont couverts (graphismes, fenêtres, langage machine) et des

super programmes sont inclus dans ce best-seller (gestion de fichiers, éditeur de texte et de sons...).

Réf. : ML112
Prix : 149 FF



PROGRAMMES BASIC POUR LE CPC 464

ALIMENTEZ VOTRE CPC 464

Ce livre contient de super programmes, notamment un

désassembleur, un éditeur graphique, un éditeur de texte... Tous les programmes sont prêts à être tapés et abondamment commentés.

Réf. : ML119
Prix : 129 FF

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, analyses des problèmes, algorithmes complexes...).

Principaux thèmes abordés :

- Les bases de la programmation

- Bit, Octet, ASCII
 - Instructions du BASIC
 - Organigrammes
 - Les fenêtres
 - Programmes BASIC plus poussés
 - Le programme et menus.
- Comprenant de nombreux exemples, ce livre vous assure un apprentissage simple et efficace du BASIC CPC 464.

Réf. : ML118
Prix : 149 FF



MICRO APPLICATION MICRO APPLICATION

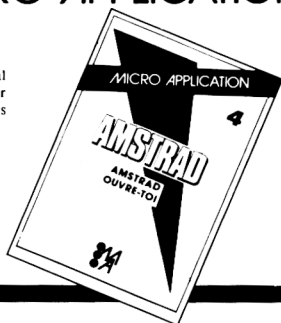
AMSTRAD OUVRE-TOI

Le bon départ avec le CPC 464 ! Ce livre vous apporte les principales informations sur l'utilisation, les possibilités de connexions 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.

Réf. : ML120

Prix : 99 FF



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 utilitaires et fichiers de jeux. Ainsi qu'un

générateur d'aventures pour programmer vous-mêmes facilement vos jeux d'aventures. Avec, bien sûr, des programmes tout prêts à être tapés.

Réf. : ML121

Prix : 129 FF

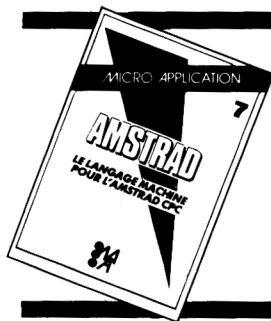
LA BIBLE DU PROGRAMMEUR DE L'AMSTRAD CPC 464 (Tome 6)

Tout, absolument tout sur le CPC 464. Ce livre est l'ouvrage de référence pour tous ceux qui veulent programmer en pro leur CPC. Organisation de la mémoire, le

contrôleur vidéo, les interfaces, l'interpréteur et toute la ROM DESASSEMBLEE et COMMENTEE sont quelques-uns des thèmes de cet ouvrage de 700 pages.

Réf. : ML122

Prix : 249 FF



LE LANGAGE MACHINE DE L'AMSTRAD CPC (Tome 7)

Ce livre est destiné à tous ceux qui désirent aller plus loin que le BASIC. Des bases de la programmation en assembleur à l'utilisation des

routines système, tout est expliqué avec de nombreux exemples. Contient un programme assembleur, moniteur et désassembleur.

Réf. : ML123

Prix : 129 FF

MICRO APPLICATION MICRO APPLICATION

MICRO APPLICATION MICRO APPLICATION

GRAPHISMES ET SONS DU CPC

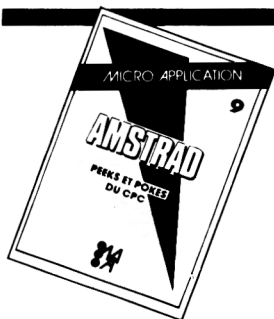
L'AMSTRAD CPC dispose de capacités graphiques et sonores exceptionnelles. Ce livre en montre l'utilisation à l'aide de nombreux programmes utilitaires.

Contenu :

- base de programmation graphique
- éditeur de police de caractères
- "sprites", "shapes", et chaînes
- représentations multi-couleurs.
- calcul des coordonnées

- rotations, mouvements
- représentations graphiques de fonctions en 3D
- D.A.O. (dessin assisté par ordinateur)
- synthétiseur
- mini-orgue
- enveloppes de son, et beaucoup d'autres choses...

Ref : ML124
Prix : 129 FF



PEEKES ET POKES DU CPC (Tome 9)

Comment exploiter à fond son CPC à partir du BASIC ? C'est ce que vous révèle ce livre avec tout ce qu'il faut savoir sur les peeks, pokes et autres call... Vous saurez aussi

comment protéger la mémoire, calculer en binaire... et tout cela très facilement. Un passage, assuré et sans douleur du BASIC au puissant LANGUAGE MACHINE.

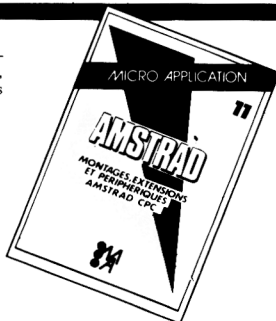
Ref : ML126
Prix : 99 FF

MONTAGES, EXTENSIONS ET PERIPHERIQUES AMSTRAD CPC (Tome 11)

Pour tous les amateurs d'électronique ce livre montre ce que l'on peut réaliser avec un CPC. De nombreux schémas et exemples

illustrent les thèmes et applications abordés comme les interfaces, programmeur d'EPROM... Un très beau livre de 450 pages.

Ref : ML131
Prix : 199 FF



MICRO APPLICATION MICRO APPLICATION

MICRO APPLICATION MICRO APPLICATION

LE LIVRE DU CP/M AMSTRAD (Tome 12)

Ce livre vous permettra d'utiliser CP/M sur les CPC 464, 664 et 6128 sans aucune difficulté. Vous y trouverez de nombreuses explications

et les différents exemples vous assureront une maîtrise parfaite de ce très puissant système d'exploitation qu'est CP/M. (300 pages).

Ref. : ML128
Prix : 149 FF



DES IDEES POUR LES CPC (Tome 13)

Vous n'avez pas d'idées pour utiliser votre CPC (464, 664, 6128) ? Ce livre va vous en donner ! Vous trouverez de très nombreux programmes BASIC couvrant des sujets très variés qui transformeront votre

CPC en un bon petit génie. De plus les programmes vous permettront d'approfondir vos connaissances en programmation. (250 pages).

Ref. : ML132
Prix : 129 FF

AMSTRAD AUTOFORMATION A L'ASSEMBLEUR EN FRANCAIS

Contient un livre et un logiciel.

LE LIVRE :

Cet ouvrage introduit le débutant à la programmation du Z80 grâce à la méthode du DR WATSON qui selon les critiques vaut son pesant d'or ! Aucune connaissance préalable n'est requise et le but du livre est d'assurer au novice un succès total. A la fin du livre les instructions du Z80 sont expliquées en détail. De nombreux exemples illustrent les différentes étapes du cours alors que des exercices (les solutions sont fournies) testent la compréhension.

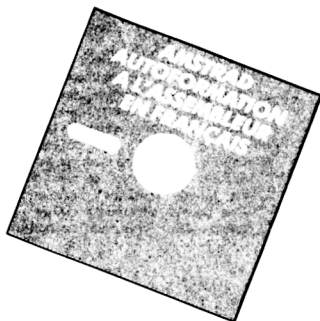
LE LOGICIEL : Un assembleur Z80

complet est livré sur cassette et comprend :

- Etiquettes Symboliques
- Directives d'Assemblage
- Chargement/Sauvegarde
- Copie Ecran
- INSERT / DELET.

L'assembleur permet d'écrire des programmes facilement en langage d'assemblage puis les transforme en code machine (langage machine). Pour vous aider à comprendre les rotations mathématiques utilisées, une démonstration de l'utilisation des nombres binaires et hexadécimaux est fournie. Un programme utilisant les commandes graphiques additionnelles décrites dans le livre est également fourni.

Ref. : ML126
Prix : 195 FF K 7 - 295 FF - disquette



MICRO APPLICATION MICRO APPLICATION

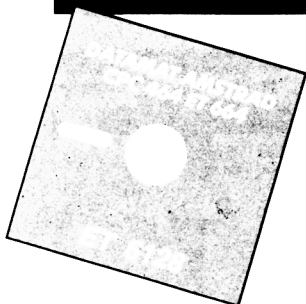
MICRO APPLICATION MICRO APPLICATION

TEXTOMAT AMSTRAD CPC 464 & 664

Traitement de texte de qualité professionnelle pour tous. Tabulation, recherche, remplacement, insertion, manipulation de paragraphes, calcul... Accents à l'écran et imprimante. Module permettant de

gérer tout type d'imprimante. Ecrit en LANGUAGE MACHINE. Liaison avec DATAMAT pour mailing et lettres types personnalisées... TEXTOMAT est la solution traitement de texte sur CPC. Documentation complète.

Réf. : AM305
Prix : 450 FF



DATAMAT AMSTRAD CPC 464 & 664

La gestion de fichier la plus complète fonctionnant pour les 464 et 664. Entièrement en LANGUAGE

MACHINE. Fonctions de calcul, de tri, de recherche multicritères, impressions paramétrables, liaison avec TEXTOMAT pour mailing... Documentation française de 60 pages.

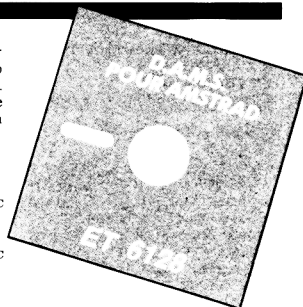
Réf. : AM304
Prix : 450 FF

D.A.M.S. POUR AMSTRAD CPC 464 & 664

D.A.M.S. est un logiciel intégrant un assembleur, un moniteur et un désassembleur symbolique pour développer et mettre au point facilement des programmes en langage machine sur les micro ordinateurs AMSTRAD. Les trois modules sont co-résidents en mémoire ce qui assure une grande souplesse d'utilisation. Vous pouvez notamment utiliser un éditeur plein écran, un assembleur

immédiat, un désassembleur symbolique, une trace et beaucoup d'autres fonctions très puissantes. D.A.M.S. est entièrement relogable et est bien évidemment écrit en langage machine.

Réf. : AM208
Prix : sur cassette : 295 FF TTC
pour CPC 464
Réf. : AM308
Prix : sur disquette : 395 FF TTC
pour CPC 664 & CPC 464



MICRO APPLICATION MICRO APPLICATION

Achévé d'imprimer en février 1986
sur les presses de l'imprimerie Laballery et C^{ie}
58500 Clamecy
Dépôt légal : février 1986
N° d'imprimeur : 602017

Ce livre s'adresse à tous ceux qui désirent aller plus loin dans l'utilisation des langages BASIC.

Cet ouvrage essaye de montrer la place importante des routines de l'Amstrad au niveau de la programmation assembleur.

Le lecteur y trouvera tout sur la structure d'un programme basic et ses variables en mémoire.

Puis, pour chaque thème (clavier, cassette, système, graphisme), sont étudiées un maximum de routines disponibles en RAM.

Une large place a été réservée au graphisme, principal atout de l'Amstrad. De nombreux exemples commentés, en assembleur, viennent illustrer les sujets traités. En annexe on trouvera quelques programmes utilitaires en BASIC, dont un duplicateur de programme.

Contenu :

- Description hardware
- Organisation de la mémoire (RAM et ROM)
- Structure d'un programme Basic en mémoire
- Gestion du clavier
- L'écran texte
- L'écran graphique
- Gestion de la mémoire écran
- L'unité de cassette
- Système
- Tables inverses des instructions du Z80
- Programmes utilitaires (duplicateur, super-dump, etc.)
- Lexique

Avec ce livre la programmation en assembleur prend une nouvelle dimension, permettant d'accéder à une réelle maîtrise de votre ordinateur.

AMSTRAD

LES ROUTINES DE L'AMSTRAD

CPC 464, 664 ET 6128



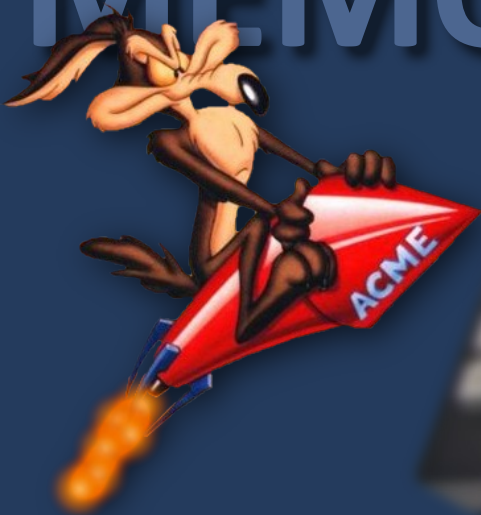


Document **numérisé**
avec amour par :

AMSTRAD

CPC 

MÉMOIRE ÉCRITE



<https://acpc.me/>