

Sixième partie

L'extramoniteur

1. Généralités

Principes de base

L'extramonitor est une bibliothèque de logiciels intégrée aux TO8, TO9 et TO9+ dont le but est de compléter les fonctions gérées par le monitor décrit précédemment. Il met à notre disposition un panel de routines bien utiles permettant de simplifier considérablement l'écriture de programmes en assembleur et d'optimiser leur temps d'exécution. Les domaines abordés sont variés, il y en a pour tous les goûts: graphisme, tortue, mathématiques, musique et gestion de disquettes.

La procédure pour utiliser des programmes de l'extramonitor est légèrement différente de celle utilisée pour le monitor:

1) On appelle l'extramonitor à l'aide du monitor en faisant un JSR à l'adresse \$ECOC.

2) Par le contenu de l'accumulateur B, on précise quel programme doit être exécuté.

Donc d'une manière générale, l'écriture sera souvent:

```
EXTRA      EQU    $ECOC
           LDB     #CODE
           JSR     EXTRA
```

Le CODE étant le numéro de la routine concernée.

L'utilisation de cette bibliothèque nécessite néanmoins quelques précautions d'emploi strictes et indispensables, que nous avons rassemblées et appelées les 5 commandements (sic!):

Premier commandement

Avant toute chose, initialiser EXTRAMON avec RESETC (à froid) ou RESETW (à chaud).

Second commandement

Ne pas avoir de sous-programme d'interruption dans l'espace \$0000-\$4000.

Troisième commandement

Si vous utilisez le disque, il faut indiquer les zones tampons grâce à FCBINI.

Quatrième commandement

Il est important de penser à gérer les erreurs déclarées par EXTRAMON. En conséquence, à la sortie de toute routine il est recommandé (surtout pour les programmeurs inexpérimentés) de tester le registre B. Dans le cas où tout s'est déroulé normalement, l'accu B contiendra 0. Dans le cas contraire, il contiendra un code correspondant aux codes d'erreur du BASIC.

Cinquième commandement

Il faut avoir en mémoire que les paramètres d'entrée sont implantés soit dans les registres du 6809 E, soit dans la zone RAM \$6100-\$62FF. Les paramètres de retour sont, eux, tous en RAM. Tous les registres, sauf B, sont préservés.

Initialisation d'extramon

Conformément au premier commandement, le premier appel d'EXTRAMON doit concerner son initialisation. Cette tâche est réalisée par deux routines:

RESETC pour un reset à froid
RESETW pour un reset à chaud.

Ces routines initialiseront la zone mémoire d'EXTRAMON et feront un test non destructif de la RAM. En retour, la variable NBANK contiendra le nombre de banques RAM présentes dans le système. Le registre MODELE désignera le type d'unité centrale selon le codage suivant:

Code dans Modèle	Unité centrale
------------------	----------------

\$01	TO9
\$02	TO8
\$03	TO9+

La densité des drives sera initialisée, le registre TYPDSK contenant le type de contrôleur présent selon la formulation suivante:

Code dans TYPDSK	Contrôleur reconnu
------------------	--------------------

\$00	Contrôleur simple densité 80K
\$01	Contrôleur double densité 5"1/4
\$02	Contrôleur double densité 3"1/2
\$03	Contrôleur QDD 50K

Nom	: RESETC
Code d'entrée	: 00
Paramètre d'entrée	: Néant
Paramètre de retour	: Registre NBANK \$618C Registre MODELE \$627B Registre TYPDSK \$6219 (TO8, TO9+)
Effet	: Initialise "à froid" la zone RAM d'EXTRAMON, retourne dans NBANK le nombre de banques RAM disponibles, dans MODELE le type d'unité centrale et dans TYPDSK le type de contrôleur.

Nom	: RESETW
Code d'entrée	: 01
Paramètre d'entrée	: Néant
Paramètre de retour	: Toute la zone RAM concernée par EXTRAMON
Effet	: Initialise "à chaud" la zone RAM d'EXTRAMON.
Exemple	:

EXTRA	ORG	\$A000
RESETW	EQU	\$EC0C
	EQU	\$01
	LDB	#RESETW
	ISR	EXTRA
	SWI	
	END	

2. Le graphique

Généralités

Avant d'appeler une routine graphique, il est impératif:

- d'appeler une fois la routine CHOIX pour choisir son mode de tracé,
- de définir sa fenêtre de travail.

La plupart des routines graphiques travaillent à partir du point de coordonnées XXXX,YYYY appelé curseur graphique et correspondant au centre des ellipses, à la première extrémité des droites, des rectangles, etc.

La fenêtre de travail

L'EXTRAMON ne détermine pas implicitement de fenêtre de travail. Afin d'afficher du graphisme à l'écran, il est donc indispensable de définir ses dimensions. La fenêtre de travail est déclarée en initialisant directement les registres XL,YB et XR,YT (XLeft, YBottom, XRight, YTop). Aucun appel à EXTRAMON n'est donc à effectuer. En conséquence, faites attention car il ne vérifie rien. Ainsi:

XL (\$61A5-\$61A6) définit la marge gauche.

YB (\$61A7-\$61A8) définit la marge haute.

XR (\$61A9-\$61AA) définit la marge droite.

YT (\$61AB-\$61AC) définit la marge basse.

Pour travailler sur l'écran complet en 40 colonnes, il faut mettre :

XL = 0, XR = 319, YB = 0, YT = 199.

Choix du type de tracé

Trois types d'écrans sont compréhensibles par l'EXTRAMON du TO9, et quatre par l'EXTRAMON des TO8 et TO9+:

Le plan 320 × 200 16 couleurs dit mode TO7-70

Le plan 640 × 200 bicolore dit 80 colonnes

Le plan 320 × 200 4 couleurs dit bit-map 4 couleurs

Le plan 160 × 200 16 couleurs dit bit-map 16 (TO8 et TO9+)

Ces plans doivent être préalablement choisis par une séquence d'échappement ESC de la routine PUTC du moniteur. La routine CHOIX permet alors de

déterminer le type de tracé utilisé par les graphismes. Les options sont les suivantes:

Contenu de TRATYP	Tracé correspondant
\$00	Tracé normal
\$01	Mode transparent (OU logique)
\$02	Mode inversion (OU exclusif)
\$03	Mode ET ('008 et '009+ uniquement)

La routine CHOIX sera sollicitée pour tout changement de mode. Le type de tracé sera implanté dans le registre TRATYP, et le graphisme prendra la couleur désignée dans le registre COULEUR (-16 à +15). Le drapeau WITH précise si le tracé sera avec la couleur de fond :

\$00	avec la couleur de fond
\$FF	sans la couleur (forme uniquement).

Dans le mode T07-T0, des couleurs négatives feront tracer en fond au lieu de forme. Si WITH est à \$FF, seul le plan mémoire FORME est affecté.

En mode bit-map 4 couleurs, seuls les deux bits de poids faible de COULEUR sont pris en compte. L'octet WITH à \$FF implique que l'écriture ne sera réalisée que dans le plan FORME (ici plan ROUGE).

En mode 80 colonnes, seul le signe de COULEUR agira sur le tracé (tracé de 1 ou de 0). L'octet WITH n'a pas d'effet.

Nom	CHOIX
Codé d'entrée	28
Paramètres d'entrée	TRATYP \$61A0 COULEUR \$61B0 WITH \$6280
Paramètre de sortie	: Néant
Effet	Détermine le type de tracé utilisé par les graphismes.

Tracé d'un point

La routine PSETXY permet d'afficher un point graphique dont les coordonnées sont exprimées dans le registre X (abscisse) et le registre Y (ordonnée) du 6809 E. En retour, les registres XXXX et YYYY contiendront les coordonnées du dernier point tracé. Précisons que le point ne sera tracé que s'il est à l'intérieur de la fenêtre de travail.

Nom	: PSETXY
Code d'entrée	: 25
Paramètres d'entrée	: Registre X du 6809 E Registre Y du 6809 E
Paramètres de sortie	: XXXX \$61A1-\$61A2 YYYY \$61A3-\$61A4
Effet	: Trace un point graphique repéré par X et Y
Exemple	: Le programme suivant affiche un point de coordonnées 100,50.

* TRACE D'UN POINT GRAPHIQUE PAR
* EXTRAMON

```

                TITLE  EXPOINT
                ORG     $A000
EXTRA EQU       $EC0C
RESETW EQU      01
PSETXY EQU      25
XL EQU         $61A5
XR EQU         $61A9
YB EQU         $61A7
YT EQU         $61AB
XXXX EQU       $61A1
YYYY EQU       $61A3

FLAG EQU        *          RESET GENERAL
                LDB      #RESETW
                JSR      EXTRA  reset extramon
                LDX      #0
                STX      XL      declaration
                STX      YB      de
                LDX      #319    la
                STX      XR      fenetre
                LDX      #199    de
                STX      YT      travail

FLAG1 EQU        *          prog etudie
                LDX      #100    colonne=100
                LDY      #50     ligne =50
                LDB      #PSETXY appel
                JSR      EXTRA
                SWI
                END

```

Tracé de droites

A l'aide de la routine LINE, vous pourrez tracer des lignes graphiques qui seront affichées si leur position est à l'intérieur de la fenêtre de travail. Le principe est de désigner les coordonnées des deux extrémités de la droite avant d'appeler la routine. L'abscisse et l'ordonnée du premier point seront respectivement implantées dans le registre XXXX et YYYY, le dernier point sera précisé dans le registre X (abscisse) et Y (ordonnée) du microprocesseur. En retour de la routine LINE, les coordonnées du dernier point tracé seront automatiquement recopiées dans les registres XXXX,YYYY.

Nom	: LINE
Code d'entrée	: 26
Paramètres d'entrée	: Registre XXXX \$61A1-\$61A2 Registre YYYY \$61A3-\$61A4 Registres X et Y du 6809 E
Paramètres de retour	: Registre XXXX Registre YYYY
Effet	: Trace une ligne graphique entre les deux points désignés par leur coordonnées.
Exemple	:

* TRACE DE DEUX LIGNES GRAPHIQUES PAR
* EXTRAMON

	TITLE	EXLINE	
	ORG	\$A000	
EXTRA	EQU	\$EC0C	
RESETW	EQU	01	
LINE	EQU	26	
XL	EQU	\$61A5	
XR	EQU	\$61A9	
YB	EQU	\$61A7	
YT	EQU	\$61AB	
XXXX	EQU	\$61A1	
YYYY	EQU	\$61A3	
FLAG	EQU	*	RESET GENERAL
	LDB	#RESETW	
	JSR	EXTRA	reset extramon
	LDX	#0	
	STX	XL	declaration
	STX	YB	de

LDX	#319	la
STX	XR	fenetre
LDX	#199	de
STX	YT	travail
MACP	EQ0	prog actuelle
LDX	#100	1 point; colon=100
LDY	#50	1 ligne=50
STX	XXXX	
STY	YYYY	
LDX	#200	2 point; colon=200
LDY	#150	1 ligne=150
LDB	#LINE	
SR	EXTRA	
LDX	#300	3 point; colon=300
LDY	#50	1 ligne=50
LDB	#LINE	
USE	EXTRA	
SWI		
END		

Motif de remplissage

Pour tout remplissage d'une figure pleine, il faut fournir un motif. Un motif est une suite de 8 octets avec les mêmes conventions que les caractères graphiques de PUTC. Ce motif est à définir en RAM et le pointeur correspondant sera implanté dans le registre MACP (\$627D-627E).

Tracé de rectangles

La routine BOX permet de tracer des rectangles en désignant les coordonnées de deux sommets opposés. La procédure d'appel est identique au tracé de droite, il suffit d'appeler BOX à la place de LINE. En conséquence, l'abscisse et l'ordonnée du premier sommet sont implantées respectivement dans XXXX et YYYY, le sommet opposé est repéré dans les registres X et Y du 6809 E. De plus, par le contenu du registre FILFLG vous indiquez si le rectangle est vide ou plein :

FILFLG = \$00 rectangle vide

FILFLG = \$FF rectangle plein (BOXP en BASIC).

Si FILFLG est à \$FF, le remplissage se fera avec un motif préalablement défini par une série de 8 octets pointés par le registre MACP (voir ci-dessus, Motif de remplissage). Le remplissage se fait dans le mode courant d'affichage choisi lors de l'appel à la routine CHOIX (normal, transparent ou inversion).

Nom
Code d'entrée
Paramètres d'entrée

BOX
27
XXXX \$61A1-\$61A2 Abaisse le sommet
YYYY \$61A3-\$61A4 Ordonne le sommet
Registre X du 6908 E Abaisse le sommet
Registre Y du 6908 E Ordonne le sommet
Registre FILELC \$61EF
Registre MACP \$627D-\$627E

Paramètres de retour

Registre XXXX
Registre YYYY

Effet

Trace le rectangle entre les deux sommets désignés
et éventuellement remplissage avec un motif

Exemple

* TRACE D'UNE HOI TE PLEINE LE MOTIF DE
* REMPLISSAGE EST IMPLANTÉ EN 15000

TITLE EQU BOX
ORG EQU 1A000
EXTRA EQU 8E00C
RESSETV EQU 01
BOX EQU 27
FILELC EQU 61EF
MACP EQU 627D
XL EQU 61A5
XR EQU 61A9
YE EQU 61A7
YT EQU 61AB
XXXX EQU 61A1
YYYY EQU 61AB

FLAG EQU * RESET GENERAL
EDB #RESSETV
JSR EXTRA reset extramou
LDX #0
STX XL déclaration
STX YE de
LDX #319 la
STX XR fenêtre
LDX #100 de
STX YT travail

FLAG1	EQU	*	prog etudie
	LDA	#\$FF	
	STA	FILFLG	box plein
	LDX	#\$B007	
	STX	MACP	posit. pointeur
	LDX	#\$B000	implantation
	CLRA		du
	LDY	#MOTIF	motif
REFAI	LDE	A,Y	en
	STB	,X+	forme
	INCA		de
	CMPA	#\$08	danier
	BNE	REFAI	
	LDX	#100	1 point:colonne=10
	LDY	#50	ligne=50
	STX	XXXX	
	STY	YYYY	
	LDX	#200	2 point:colonne=200
	LDY	#150	ligne=150
	LDE	#BOX	
	JSR	EXTRA	trace de la boîte
	SWI		
MOTIF	FCB	\$AA,\$55,\$AA,\$55,\$AA,\$55	
	FCB	\$AA,\$55	
	END		

Tracé d'ellipse

Le centre de l'ellipse est le curseur graphique dont les coordonnées sont exprimées dans le registre XXXX (abscisse) et YYYY (ordonnée). Les rayons horizontaux et verticaux sont initialisés respectivement dans les registres AXEH et AXEV. En appelant la routine CIRCLE vous aurez alors le tracé d'une ellipse complète.

Mais vous avez également la possibilité de dessiner des "canemberts" (arcs d'ellipses) en mettant la valeur \$FF dans le registre CAMLG. Il faut alors donner en radian les 2 angles de l'arc dans ALPHA1 et ALPHA2 (4 octets chacun). Le registre FILFLG a ici le même sens que dans BOX.

Nom : CIRCLE
 Code d'entrée : 24
 Paramètres d'entrée : Registre XXXX \$61A1-\$61A2
 YYYY \$61A3-\$61A4
 AXEH \$61F1)
 AXEV \$61F0)
 FILFLG \$61EF
 CAMFLG \$61F2
 ALPHA1 \$61F3-\$61F6
 ALPHA2 \$61F7-\$61FA
 MACP \$627D-\$627E
 Paramètre de retour : Néant
 Effet : Dessine une ellipse vide ou pleine, totale ou partielle (camembert).

Exemple

* TRACE D'UNE ELLIPSE PLEINE. LE MOTIF DE
 * REMPLISSAGE EST IMPLANTÉ EN 38000

	TITLE	EXELIPS
	ORG	\$A000
EXTRA	EQU	\$EC0C
RESETW	EQU	01
CIRCLE	EQU	24
AXEH	EQU	\$61F1
AXEV	EQU	\$61F0
CAMFLG	EQU	\$61F2
FILFLG	EQU	\$61EF
MACP	EQU	\$627D
XL	EQU	\$61A5
XR	EQU	\$61A9
YB	EQU	\$61A7
YT	EQU	\$61AB
XXXX	EQU	\$61A1
YYYY	EQU	\$61A3

FLAG	EQU	*	RESET GENERAL
	LDB	#RESETW	
	JSR	EXTRA	reset extramon
	LDX	#0	
	STX	XL	declaration
	STX	YB	de
	LDX	#310	la
	STX	XR	fenetre
	LDX	#199	de
	STX	YT	travail

```

*EAGI EQU * ; prog. etudie
LDA #0F ; ellipse pleine
STA FILLEG
CLR A ; ellipse complete
STA CAMPLG
LDY #0007 ; point. pointeur
STA MACP ; implantation
LDY #0000 ; du
CLR A ; motif
LDY #MOIIF ; en
RSTAI LDB A,X ; forme
STH X,Y ; de
INCA ; damier
ONCA #008
BNE RSTAI
LDX #160 ; centre : colon=160
LDY #100 ; ligne=100
STX XXXX
STY YYYY
LDA #100 ; axe horizont=100
STA AXEH
LDA #50 ; axe vertical=50
STA AXEV
LDB #CIRCLE
JSR EXTRA ; trace de l'ellipse
SVI
MOVI PCB, $AA,$55,$AA,$55,$AA,$55
PCB, $AA,$55
END

```

Remplissage d'une zone

Le curseur graphique (XXXX,YYYY) est l'origine du remplissage. Toute la zone de la fenêtre connexe de même couleur que le point (XXXX,YYYY) sera peinte à l'appel de la routine PAINT. La couleur de remplissage sera celle définie par le dernier appel à la routine CHOI. Le motif de remplissage est pointé par le registre MACP (\$627D-\$627F) comme dans le cas des routines BOX et CIRCLE.

Il est indispensable, pour réaliser cette opération, de fournir une zone de travail initialisée entre DEBZON (début de zone) et FINZON (fin de zone). Si cette place est insuffisante pour la complexité du dessin, une erreur "OUT OF MEMORY" sera générée.

Nom	: PAINT
Code d'entrée	: 29
Paramètres d'entrée	: XXXX \$61A1-\$61A2 YYYY \$61A3-\$61A4 DEBZON \$616B-\$616C FINZON \$616E-\$616F MACP \$627D-\$627E
Paramètre de sortie	: Néant
Effet	: Permet de peindre une surface avec un motif de son choix.

Le Micro-Interpréteur Graphique MIG

Le micro-interpréteur graphique MIG est un outil vraiment extraordinaire dans la mesure où il intègre dans une même routine différentes fonctions étudiées précédemment. La routine MIG permet en effet de tracer des figures complexes en un minimum d'instructions. Les programmes sont ainsi plus simples à écrire, plus courts en mémoire et plus rapides en exécution.

Après avoir choisi votre type de tracé par la routine CHOIX ainsi que votre fenêtre d'affichage, MIG est capable de tracer des points, des droites, des rectangles et des ellipses (pleins ou vides).

Il y a deux manières d'utiliser cette routine:

- le macro-assembleur
- l'assembleur.

Dans les deux cas, les codes à utiliser sont du type code opération suivi d'opérandes. La plupart des commandes de MIG sont doublées de façon à pouvoir travailler sur des opérandes de 8 ou 16 bits, ceci afin de gagner de la place et du temps. Les commandes 16 bits commencent par L (pour Long). Quel que soit votre mode de traitement, voici la liste des macros graphiques de MIG :

REL	Passé en relatif
ABS	Passé en absolu (ces 2 commandes permettent de passer des coordonnées de manière relative ou absolue, au choix...)
LFCUR x,y	Positionne le curseur graphique en x,y, x,y sont ici sur 2 octets chacun.
FCUR x,y	Identique à LFCUR, mais x,y sont sur un octet chacun seulement. Utile pour les petits déplacements.

LPOIN x,y	Ecrit un point en x,y
POIN x,y	Idem mais x,y sur 8 bits
LLINE x,y	Trace une ligne jusqu'en x,y
LINE x,y	Idem mais 2 x 8 bits
LBOX x,y	Trace un rectangle
BOX x,y	Idem mais 2 x 8 bits
LBOXF x,y	Trace un rectangle plein
BOXF x,y	Idem mais 2 x 8 bits
ROND a,b	Trace une ellipse de rayon horizontal a et de rayon vertical b. Les 2 rayons sont des octets.
RONDF a,b	Idem mais l'ellipse est pleine.
MOTIF ad	Fournit un nouveau motif de remplissage: ad pointe sur les 8 octets décrivant le motif.
RMOTIF ad	Idem mais permet d'adresser le motif de manière relative au compteur de programme de MIG. Ceci permet de générer des suites de commandes relogeables.
LRMOTIF ad	Idem à RMOTIF mais en relatif 16 bits.

En plus des commandes graphiques, vous disposerez de quelques instructions de contrôle.

CALL ad	Permet l'appel à un sous-programme. Ce sous-programme est bien entendu écrit lui aussi en MIG et doit se terminer par STOP qui est un retour de sous-programme.
RCALL ad	Idem mais relatif au compteur de programme de MIG
LRCALL ad	Idem mais relatif 16 bits.
DO n	Permet de répéter n fois la séquence qui suit jusqu'au LOOP associé.
MULTIP n	Permet de multiplier l'appel à une fonction. Si vous devez appeler 18 fois de suite LINEC, il vaut mieux appeler: MULTIP 18 LINEC x1,y1, x2,y2,..., x18,y18 L'économie ici réalisée est de 15 octets...

Dans le cas où vous travaillez avec un macro-assembleur, le tableau ci-après donne le listing des codes à planter en FCB pour créer les macro-instructions.

Instructions de l'interpréteur graphique M.I.G.

STOP	MACRO		Retour sous programme
	FCB	0	ou principal
	ENDM		
CALL	MACRO	ARG	Appel de sous programme
	FCB	1	
	FDB	ARG	
	ENDM		
RCALL	MACRO	ARG	BSR
	BRA	ARG	
	ORG	*-2	
	FCB	19	
	FCB	ARG*-1	
LRCALL	MACRO	ARG	LBSR
	FCB	20	
	FDB	ARG*-2	
	ENDM		
DO	MACRO	ARG	Structure de contrôle
	FCB	2,ARG	
	ENDM		
LOOP	MACRO		Le complément du DO
	FCB	0	
	ENDM		
MULTIP	MACRO	ARG	Cette macro est généralement
	FCB	3,ARG	suivie d'une suite de FCB
	ENDM		
MOTIF	MACRO	ARG	Pointeur vers un pattern
	FCB	4	
	FDB	8+(ARG)	
	ENDM		
RMOTIF	MACRO	ARG	
	BRA	ARG+8	
	ORG	*-2	
	FCB	21	
	FCB	8+(ARG)*-1	
	ENDM		
LRMOTIF	MACRO	ARG	
	FCB	22	
	FDB	8+(ARG)*-2	
	ENDM		
ABSOLU	MACRO		Passer en absolu
	FCB	5	
	ENDM		

RELATIF	MACRO FCB ENDM	6	Passé en relatif
LCUR	MACRO FCB FDB ENDM	ARGX, ARGY 7	Positionne le curseur
FCUR	MACRO FCB ENDM	ARGX, ARGY 8, ARGX, ARGY	
LPOIN	MACRO FCB FDB ENDM	ARGX, ARGY 9	Ecrit un point
POIN	MACRO FCB ENDM	ARGX, ARGY 10, ARGX, ARGY	
LLINE	MACRO FCB FDB ENDM	ARGX, ARGY 11	Trace une ligne
LINE	MACRO FCB ENDM	ARGX, ARGY 12, ARGX, ARGY	
LBOX	MACRO FCB FDB ENDM	ARGX, ARGY 13	Trace un rectangle
BOX	MACRO FCB ENDM	ARGX, ARGY 14, ARGX, ARGY	
LBOXF	MACRO FCB FDB ENDM	ARGX, ARGY 15	Trace un rectangle plein
BOXF	MACRO FCB ENDM	ARGX, ARGY 16, ARGX, ARGY	
ROND	MACRO FCB ENDM	ARGX, ARGY 17, ARGX, ARGY	Trace une ellipse
RONDF	MACRO FCB ENDM	ARGX, ARGY 18, ARGX, ARGY	Une ellipse pleine

Pour les moins fortunés (comme nous) qui ne possèdent pas de macro-assembleur, pas de panique! Le MIG peut également être utilisé. Dans le listing ci-dessus vous voyez qu'à chaque instruction de MIG correspond un code en FCB:

- pour ROND c'est 17
- pour STOP c'est 0
- pour BOXF c'est 16 etc.

Bien! Dans le tableau précédent (liste des macros graphiques de MIG) sont indiquées les syntaxes de chaque instruction (nombre d'octets et cadrage). En assembleur, il suffira de rentrer une figure graphique constituée de séquences écrites en FCB pour travailler avec MIG.

Exemple d'application: On veut tracer une figure complexe constituée:

- d'une boîte (remplie) de coordonnées 150,50 et 200,100
- d'un cercle de coordonnées 200,100 et 50,50
- d'un segment de droite de coordonnées 50,50 à 150,150

Nous écrivons donc la séquence d'octet suivante:

- 06 pour travailler en relatif
- 16,200,100 pour afficher une boîte pleine selon coordonnées
- 17,50,50 pour afficher un cercle selon les coordonnées 50,50
- 12,150,150 pour tracer la ligne jusqu'à 150,150
- 0 pour arrêter la séquence MIG (indispensable).

Dans sa totalité le programme peut s'écrire de la manière suivante:

```
* TRACE D'UNE BOITE PLEINE, D'UN CERCLE
* VIDE ET D'UNE DROITE PAR LE M.I.G
```

	TITLE	EXMIG
	ORG	\$A000
EXTRA	EQU	\$EC0C
RESETW	EQU	01
MIG	EQU	30
XL	DBQ	\$61A5
XR	EQU	\$61A9
YB	EQU	\$61A7
YT	EQU	\$61AD
XXXX	EQD	\$61A1
YYYY	EQU	\$61A3

```

FLAG      EQU      *          RESET GENERAL
          LDB      #RESETW
          JSR      EXTRA      reset extramem
          LDX      #0
          STX      XL          declaration
          STX      YB          de
          LDX      #319        la
          STX      XR          fenetre
          LDX      #199        de
          STX      YT          travail

FLAG1     EQU      *          prog etudie
          LDX      #150        positionnement
          STX      XXXX        du
          LDX      #50         curseur
          STX      YYYY        graphique
          LDX      #FIGURE     charge graphisme
          LDB      #MIG         appel a MIG
          JSR      EXTRA
          SWI

FIGURE    FCB      05          mode relatif
          FCB      16,200,100 BOXF (200,100)
          FCB      17,50,50   ROND
          FCB      12,150,150 LINE (150,150)
          FCB      $0          stop

          END

```

Si vous travaillez en macro-assembleur (petits veinards) la différence essentielle se situe sur la partie en flag FIGURE qui s'écrira:

```

FIGURE    REL
          BOXF      200,100
          ROND      50,50
          LINE      150,150
          STOP

```

Les néophytes auront certainement remarqué au passage qu'un programme écrit à l'aide des macro instructions a des allures de BASIC.

Nom	: MIG
Code d'entrée	: 30
Paramètres d'entrée	: Registre XXXX \$61A1-\$61A2 Registre YYYY \$61A3-\$61A4 Registre X du 6809 E
Paramètre de sortie	: Néant
Effet	: A partir du curseur graphique de coordonnées XXXX et YYYY, trace une figure graphique définie en FCB et pointée par le registre d'index X.
Exemple	: voir le programme EXMIG ci-avant.

Codage et décodage d'images

Codage

Nom	: CODE		
Code d'entrée	: 69 (pour TO9+ et TO8)		
Paramètres d'entrée	:		
PUTFLG	\$6249		\$00
X0COD	\$61D6, Y0COD	\$61D7	Coin supérieur gauche
X1COD	\$61D8, Y0COD	\$61D9	Coin inférieur droit
DERZON	\$616B-\$616D		Début de la zone résultat
WITH	\$6288		\$00 ==> avec la couleur
			\$FF ==> sans la couleur
PASSCD	\$61DB		\$00 ==> codage virtuel
			\$FF ==> code, rangé en mémoire
Paramètres de retour	: LSTBYT (\$61DC-\$61DE) Le 1er octet libre après le codage.		
Effet	: Permet de coder un dessin. Les coins de l'image sont donnés en coordonnées caractères. La zone mémoire est remplie si PASSCD est à \$FF, sinon, seul le registre LSTBYT est mis à jour. Ceci permet de tester si une zone mémoire est suffisante pour coder l'image de l'écran.		

Décodage

Nom	CODE
Code d'entrée	: 69 (pour TO24 et TO8)
Paramètres d'entrée	
PUTFLG	\$6249 \$FF
DEBZON	\$616B-\$616D Adresse début du dessin
XCCOD	\$661D6 YCCOD \$61D7 Coin supérieur gauche
WITH	\$6288 \$00 ==> avec la couleur
	\$FF ==> sans la couleur
Paramètre de sortie	Néant
Effet	: Permet de décoder une image. Le troisième octet de DEBZON contient le numéro de banque logique (de 1 à 6). Si le coin supérieur gauche est trop à droite, ou trop bas pour que le dessin soit affiché, rien n'est tracé. Le décodage se fait dans le mode choisi par la routine CHOIX et permet donc le "ou exclusif" ainsi que d'autres subtilités (voir CHOIX).

3. Les tortues

Généralités

Une tortue est un objet graphique (fil de fer) que l'on peut déplacer, agrandir, déformer à volonté. D'une manière générale, pour parler à une tortue en appelant EXTRAMON, il suffit d'implanter un code dans l'accumulateur B correspondant à la fonction désirée et de fournir un pointeur vers le descripteur de la tortue dans le registre Y du 6809 E.

Ce descripteur a une longueur de $14 + n$ octets, n dépendant de la complexité de la forme de la tortue. La plupart des actions réalisables avec une tortue se font grâce à un appel d'EXTRAMON. La manipulation des commandes est très proche de celle du BASIC 128.

Initialisation

Première étape, initialiser la tortue. Cette opération est réalisée par la routine INITORTUE. Après exécution de cette routine:

- La zone est initialisée.
- La tortue est au centre de l'écran, invisible, crayon levé.
- Echelle normale, direction et rotation nulles, mode "rapide".
- Elle est en forme de triangle isocèle (un peu comme la tortue LOGO).

Nom	: INITORTUE
Code d'entrée	: 39
Paramètre d'entrée	: Registre Y du 6809E
Paramètre de sortie	: Néant
Effet	: Initialise une tortue. Le registre Y pointant une zone de 27 octets au minimum.
Exemple	: Voir programme EXTORTUE page 261.

La visibilité

La routine SHOW affiche ou cache une tortue désignée par le registre d'index Y du 6809 E. Les tortues sont dessinées en mode "OU exclusif" afin que ces opérations soient rapides. La couleur de la tortue est choisie par la routine CHOIX.

En mode TO170, la couleur n'est pas mise en vidéo inverse. Ainsi une tortue peut laisser des traces de couleurs. Vous pourrez éviter cet effet également par la routine CHOIX.

Nom	: SHOW
Code d'entrée	: 33
Paramètres d'entrée	: Registre Y du 6809 E Accumulateur A du 6809 E
Paramètre de sortie	: Néant
Effet	: Si A = \$00 la tortue désignée par Y s'efface Si A = \$FF la tortue désignée par Y s'affiche
Exemple	: Voir programme EXTORTUE page 261.

Le déplacement

La routine FWD permet de déplacer la tortue désignée par le registre Y dans la direction courante. Ceci correspond à la commande AVANCE de l'interpréteur LOGO. Le déplacement a lieu dans une fenêtre virtuelle sphérique de - 32768 à 32767 en X comme en Y. La partie visible étant définie par la fenêtre de visualisation. Le pas de déplacement compris entre -256 (recul) et +256 (avance) est à préciser dans le registre d'index X.

Nom	: FWD
Code d'entrée	: 37
Paramètres d'entrée	: Registre Y du 6809 E Registre X du 6809 E
Paramètre de sortie	: Néant
Effet	: Déplace une tortue
Exemple	: Voir programme EXTORTUE page 261.

La direction

Les angles de direction sont fournis en 256ème de cercle au registre d'index X, sachant que le sens positif est celui des aiguilles d'une montre. Les actions FWD agiront dans cette nouvelle direction.

La tortue concernée est désignée par le registre d'index Y. Par l'accumulateur A, on précise si l'angle est absolu (A = 00) ou relatif (A = \$FF). En relatif, ceci correspond à l'instruction DROITE de l'interpréteur LOGO. En absolu, cela correspond à l'ordre FCAP (fixe cap).

Nom	: HEAD
Code d'entrée	: 34
Paramètres d'entrée	: Registre Y du 6809 E Registre X du 6809 E Accumulateur A du 6809 E
Paramètre de sortie	: Néant
Effet	: Détermine la direction pour le futur déplacement de la tortue.
Exemple	: Voir programme EXTORTUE page 261.

La rotation

Le sens positif est le sens des aiguilles d'une montre. Les angles sont fournis en 256ème de cercle. La forme de la tortue désignée par le registre Y tourne sur elle-même de X 256ème de cercle sans changer de direction. L'angle est écrit dans le registre X, l'accu A précise s'il est absolu (A = 00) ou relatif (A = \$FF).

Nom	: ROT
Code d'entrée	: 35
Paramètres d'entrée	: Registre Y du 6809 E Registre X du 6809 E Accu A du 6809 E
Paramètre de sortie	: Néant
Effet	: La tortue désignée par Y pivote d'un angle écrit dans X.
Exemple	: Voir programme EXTORTUE page 261.

La taille

Nous avons la possibilité de modifier les dimensions de la tortue et de créer ainsi de véritables zooms avant, d'où l'appellation de cette routine. L'agrandissement est limité de manière absolue entre 0 et 255.

Il est conseillé de faire attention sur les agrandissements d'objets. En effet, pour des questions de rapidité EXTRAMON ne teste rien. Si un segment d'un objet dépasse 256 pixels, il peut être mal affiché.

Comme pour les autres routines, le registre d'index Y désigne la tortue concernée, le registre d'index X contient la variable (ici la valeur du zoom), et l'accumulateur A indique si l'agrandissement est absolu (A = 00) ou relatif (A = \$FF).

Nom	: ZOOM
Code d'entrée	: 36
Paramètres d'entrée	: Registre Y du 6809 E Registre X du 6809 E Accu A du 6809 E
Paramètre de sortie	: Néant
Effet	: Les dimensions de la tortue désignée par Y sont modifiées selon le rapport écrit dans X et A.

La trace

A chaque tortue est associé un crayon qui peut ainsi laisser une trace de ses déplacements comme le font les pneus boueux d'une voiture sur le bitume (!). Habilement maniée, cette fonction permet de dessiner comme avec LOGO.

La routine TRACE demande à la tortue pointée par le registre d'index Y de baisser son crayon (A = SFF) ou de le relever (A = \$00). En LOGO, la différence est faite par les primitives BC (Baisse Crayon) et LC (Lève Crayon). Le tracé se fait dans le mode courant choisi avec la routine CHOIX.

Nom	: TRACE
Code d'entrée	: 31
Paramètres d'entrée	: Registre Y du 6809 E Accu A du 6809 E
Paramètre de sortie	: Néant
Effet	: La tortue pointée par Y laisse une trace de ses déplacements ou non, en fonction de l'accu A.

La vitesse

La routine ANIME permet de moduler la qualité de l'affichage en donnant priorité à la vitesse ou au suivi de l'affichage. Deux modes sont en effet à notre disposition:

- le mode lent
- le mode rapide

La différence essentielle est qu'en mode lent on réaffiche la tortue à chaque modification, alors qu'en mode rapide EXTRAMON ne réaffiche que lors d'un déplacement de la tortue.

En particulier, les actions de ZOOM et de ROT sont différées au prochain mouvement de l'objet. Ceci permet de modifier plusieurs paramètres de la tortue assez rapidement.

On peut aussi, pour des questions de vitesse ou pour déplacer la tortue autrement que par FWD, modifier directement certains octets du descripteur. Les modifications étant faites, il faut appeler MOVE (chapitre suivant) pour les voir à l'écran. De cette manière, en un seul appel à EXTRAMON, nous pouvons changer la taille, la position, la rotation, etc. de la tortue.

Nom	: ANIME
Code d'entrée	: 32
Paramètres d'entrée	: Registre Y du 6809 H Accu A du 6809 E
Paramètre de sortie	: Néant
Effet	: La tortue pointée par Y travaillera soit en mode lent (A = \$FF) soit en mode rapide (A = \$00).

Le positionnement

Pour modifier la forme d'une tortue, il faut changer la forme standard proposée par EXTRAMON. Sauf si vous êtes amateur de sensations fortes du style "transmutations corporelles" (bonsoir docteur!), il est conseillé de demander à la tortue de se cacher avant de modifier sa forme (un peu comme superman) pour éviter de bizarres effets sur votre écran.

Cette forme se trouve dans le descripteur à partir de TFORME (Y +16). En fait, la forme d'une tortue est une série de commandes consistant à avancer, tourner, lever ou baisser le crayon. Elle est donc définie par une série de doublets ou de triplets.

Exemple:

TFORME	FDB	0,15,0	LC AV 15 TD 0 (BC)
	FDB	30,117	AV 30 TD 117
	FDB	15,74	AV 15 TD 74
	FDB	30,74	AV 30 TD 74
	FDB	0,0	

En général, ce seront des doublets (LONGUEUR, ANGLE), mais notons que si la longueur vaut zéro, alors le doublet qui suit sera effectué crayon levé. Deux zéros terminent la description. L'exemple donné ci-dessus correspond à la forme attribuée par défaut aux tortues.

Voici la liste des paramètres modifiables dans le descripteur:

TX	(en Y + 6)	Abcisse de la tortue	(3 octets)
TY	(en Y + 9)	Ordonnée de la tortue	(3 octets)
TROT	(en Y + 12)	Rotation	(1 octet)
TTAI	(en Y + 13)	Taille	(1 octet)
TDIR	(en Y + 14)	Direction	(1 octet)

TX et TY sont connus au 256ème de pixel, le dernier octet représentant la partie fractionnaire de pixel.

Nom	: MOVE
Code d'entrée	: 38
Paramètre d'entrée	: Registre Y du 6809E
Paramètre de sortie	: Néant
Effet	: Permet de modifier des paramètres de définition des tortues pointées par Y.

La compilation d'une forme

Nom	: CMPTORTUE
Numéro du point d'entrée	: 40
Paramètres d'entrée	: Registre Y du 6809 E, pointe sur la chaîne à compiler FACLO (\$6151), longueur de la chaîne de caractères Registre X du 6809 E, pointeur sur une zone de rangement
Paramètres de retour	: Registre FACLO \$6151, longueur du résultat

La syntaxe est celle utilisée dans l'ordre **TURTLE** de **BASIC 128** ou de **BASIC 512**. Le résultat de la compilation peut immédiatement être interprété par les commandes tortue.

Exemple d'une tortue en mouvement

* DEPLACEMENT D'UNE TORTUE.

	TITLE	EXTORTUR	
	ORG	\$A000	
EXTRA	EQU	\$EC0C	
RESETW	EQU	01	
INITOR	EQU	39	
SHOW	EQU	33	
FWD	EQU	37	
HEAD	EQU	34	
ROT	EQU	35	
XL	EQU	\$61A5	
XR	EQU	\$61A9	
YT	EQU	\$61AB	
YB	EQU	\$61A7	
FLAG	EQU	*	RESET GENERAL
	LDB	#RESETW	
	JSR	EXTRA	reset extramon
	LDX	#0	
	STX	XL	declaration
	STX	YB	de
	LDX	#319	la
	STX	XR	fenetre
	LDX	#199	de
	STX	YT	travail
FLAG1	EQU	*	prog etudie
	LDY	#DTOR	designie tortue
	LDB	#INITOR	initialisation
	JSR	EXTRA	
	LDA	#319	
	LDB	#SHOW	montre la tortue
	JSR	EXTRA	
REC	JSR	TEMPO	temporisation

	LDX	#15	angle=15
	LDA	#\$FF	angle relatif
	LDB	#ROT	tortue pivote..
	JSR	EXTRA	sur elle meme
	LDX	#15	angle direction=15
	LDA	#\$FF	angle relatif
	LDB	#HEAD	tortue change..
	JSR	EXTRA	de direction
	LDX	#15	avance de 15
	LDB	#FWD	deplacement tortue
	JSR	EXTRA	
	BRA	REC	on recommence
TEMPO	EQU	*	ajuster la vitesse
	LDX	#\$0FFF	par contenu de X
BOUC	LEAX	1,X	
	BNE	BOUC	
	RTS		
DTOR	EQU	*	
	END		

4. Les mathématiques

Généralités

L'EXTRAMON sait même compter! L'ensemble de ses possibilités mathématiques sont exposées dans les chapitres suivants. Passé l'effet de surprise, inmanquable l'orsque l'on apprend qu'il traite les quatres opérations, on plonge dans une béatitude inénarrable en découvrant que les fonctions trigonométriques ne sont pas épargnées (SIN, COS, TAN, ATN...), ainsi que les logarithmes népériens, exponentiel, racine carrée, conversion de bases, etc. Bref, avis aux amateurs, vous avez dans les mains une petite merveille (non, pas le livre, l'unité centrale!)

Toutes ces fonctions mathématiques seront utilisées en relation de registres appelés accumulateurs FAC et ARG. Globalement vous ferez des opérations unaires et binaires. Dans le premier cas l'argument doit se trouver dans l'accumulateur FAC, un appel à SIN ou COS, par exemple, vous rend le résultat ainsi que dans FAC. Dans le second cas, l'argument gauche doit être dans ARG et l'argument droit dans FAC. Les deux arguments seront de même type et précisés dans VALTYP. Le résultat recherché sera dans FAC. Il est conseillé de faire attention car le type du résultat peut être différent du type des opérandes. Surveillez donc bien le registre VALTYP. Si vous désirez faire des calculs en double précision, il faut nécessairement que VALTYP (\$6105) soit égal à 8, et que DBLFLG (\$6103) soit à SPH.

Description des accumulateurs

Pour fixer rapidement les idées, nous pouvons dire que FAC est l'accumulateur principal, et ARG le secondaire. ARG sert pour les opérations telles que additions, soustractions, multiplications et divisions.

Les réels courts sont codés (sauf signe) sur 4 octets. Le 1er octet est l'exposant E, les 3 autres la mantisse M.

Exposant - 128

nombre = 0.Mantisse $\times 2^E$

Exemple: Le nombre 100 décimal s'écrit:

0.C80000 $\times 2^7$

et se code en:

Exposant = $128 + 7 = 135 = \%10000111$

Mantisse = $\$C80000 = \%11001000\ 00000000\ 00000000$

Pour ces routines, un exposant nul implique que le nombre est égal à 0. En entrée, le nombre doit être normalisé (bit de poids fort de la mantisse = 1). En sortie, EXTRAMON rend également des nombres normalisés.

- Vous trouverez ci-dessous la liste des composants de FAC :

Pour un réel court ou simple précision:

VALTYP (\$6105) : type de l'accu (à 4)

FACEXP (\$614E) : exposant binaire (de - 128 + 127)

FACHO (\$614F) : mantisse 24 bits, poids forts

FACMO (\$6150) : mantisse poids moyens

FACLO (\$6151) : mantisse poids faibles

FACSGN (\$6156) : signe de l'accumulateur

Pour un réel long, c'est identique sauf:

VALTYP (\$6105) : à 8

(\$6152-\$6155) : 4 octets supplémentaires suivent la mantisse

Pour un entier 16 bits:

VALTYP (\$6105) : à 2

FACMO, FACLO contiennent l'entier

FACEXP, FACHO, FACSGN sont inutiles.

- Vous trouverez ci-dessous la liste des composants de ARG. Elle est très semblable à FAC:

ARGEXP, ARGHO, ARGMO, ARGLO (\$6159-\$615C)

DARGHO	(\$615D-\$6160)	pour les réels courts (4)
ARGSGN	(\$6161)	+4 octets pour les réels longs (8)
ARGMO, ARGLO	(\$615B-\$615C)	signe de l'argument pour réels
		pour les entiers

Dans FACSGN et ARGSGN seul le bit 7 est utilisé. Ce bit à 1 représente un nombre négatif.

Echanges mémoire et accumulateur

Les réels courts (4) sont codés sur 5 octets dans l'accumulateur, et sur 4 octets en mémoire. Ceci est possible en codant le signe du nombre dans le bit de poids fort de la mantisse, ce bit étant un 1 pour un réel normalisé. Pour les réels longs, on agit de même, mais avec 4 octets de plus pour la mantisse. Les routines fonctionnent aussi pour les entiers, X pointant alors sur 2 octets seulement.

Pour utiliser les routines de transfert bi-directionnelles entre mémoire et accumulateur:

- il faut implanter le type de la variable dans VALTYP,
- puis appeler une des routines de transfert avec le registre d'index X du 6809 E pointant sur la variable à transférer. Soit:

MOVFM	Code d'entrée 62;	Transfert de FAC vers la mémoire pointée par X.
MOVMF	Code d'entrée 63;	Transfert de la mémoire pointée par X vers l'accumulateur FAC
MOVAF	Code d'entrée 64;	Transfert de l'accumulateur ARG vers l'accumulateur FAC.

Liste des fonctions mathématiques

Le tableau ci-dessous dresse la liste des routines mathématiques disponibles et leur code d'entrée respectif.

Nom	Code d'entrée	Fonctionnalité
SGN	41	Rend le signe de l'accumulateur dans FACMO, FACLO. L'accumulateur "entier" vaut ainsi 0, -1 ou 1
INT	42	Rend la partie entière de l'accumulateur.
ABS	43	Rend la valeur absolue de l'accumulateur.
SQR	44	Rend la racine carrée de l'accumulateur. Le résultat est un nombre réel.
LOG	45	Rend le logarithme népérien de l'accumulateur. Le résultat est un nombre réel.
EXP	46	Rend l'exponentielle de l'accumulateur. Le résultat est un nombre réel.

Nom	Code d'entrée	Fonctionnalité
COS	47	Rend le cosinus de l'accumulateur. L'accumulateur est en radians. Le résultat est un nombre réel.
SIN	48	Rend le sinus de l'accumulateur. L'accumulateur est en radians. Le résultat est un nombre réel.
TAN	49	Rend la tangente de l'accumulateur. L'accumulateur est en radians. Le résultat est un nombre réel.
ATN	68	Rend l'arctangente de l'accumulateur. Le résultat est un nombre réel (TO8 et TO9+ uniquement).
FRCTYP	50	Conversion de type. Paramètres d'entrée: - VALTYP le type courant de l'accumulateur FAC - si registre A = 2 résultat entier 2 - si registre A = 4 résultat réel 4 - si registre A = 8 résultat réel 8
FIXER	51	Rend l'entier tronqué d'un réel. Le résultat est un réel
RND	52	Rend un réel court aléatoire (4)
NEGGO	53	Effectue: $FAC = - FAC$
ADDGO	54	Effectue: $FAC = ARG + FAC$
SUBGO	55	Effectue: $FAC = ARG - FAC$
MULTGO	56	Effectue: $FAC = ARG * FAC$
DIVGO	57	Effectue: $FAC = ARG / FAC$. Les arguments sont des réels 4 ou 8 octets
EXPGO	58	Effectue: $FAC = ARG ^ FAC$. Les arguments sont des réels 4 ou 8
IMODO	59	Effectue: $FAC = ARG \text{ MOD } FAC$. Les opérandes doivent être des entiers
IDIVO	60	Effectue: $FAC = ARG$ division entière par FAC. Les opérandes doivent être des entiers
FIN	65	Conversion d'une valeur ASCII en binaire

Paramètres d'entrée:

- registre Y du 6809 E pointe sur ASCII fini par \$00

Paramètres de retour:

- FAC contient le nombre

- VALTYP le type du nombre

Les conventions sont similaires à celles de BASIC.

FIN accepte aussi les constantes binaires, octales ou hexadécimales.

PUFOUT 66

Conversion d'une valeur binaire en ASCII décimal.

Paramètres d'entrée:

- FAC (\$614E) le nombre à convertir

- VALTYP (\$6105) le type de celui-ci

- Registre Y du 6809 E pointe sur un tampon

- PUMASK (\$617C) Flag de PRINTUSING

Le flag PUMASK permet d'utiliser le PRINTUSING comme BASIC. Si ce flag est nul, le format de sortie sera choisi par PUFOUT, sinon les octets DPWID et FLDWID permettent de choisir le nombre de chiffres avant et après le point décimal.

DPWID (\$617A) nombre de chiffres après le point décimal

- FLDWID(\$617B) idem mais avant le point

PUMASK (\$617C) bit 0 notation scientifique

bit 2 signe après le nombre

bit 3 force le "+" pour les

positifs

bit 5 remplit d'" " au lieu

d'espaces

bit 7 USING or not USING

Paramètre de retour (pour le TO9):

- Le registre Y du 6809 E pointe sur l'ASCII

Pour récupérer les codes ASCII, il faut sauter les caractères indésirables (inférieurs ou égaux à \$20), ensuite vous trouverez votre nombre se terminant par un zéro binaire.

Paramètre de retour (pour les TO8 et TO9+):

- Le registre FACMO pointe sur l'ASCII.

Le registre FACMO pointe sur l'ASCII dans le tampon que vous avez fourni à travers le registre Y. Le nombre initialement implanté dans FAC est détruit après conversion.

Conversion d'une valeur binaire en ASCII
hexadécimal ou octal.

Paramètres d'entrée:

- FACMO (\$6150-\$6151) le nombre à convertir en entier 2
- Registre Y du 6809 E pointe sur le buffer résultat
- Accu A du 6809 E si \$00 la sortie est octale si SFF la sortie est hexa.

5. Le DOS

Pour toutes informations relatives aux disquettes elles mêmes, nous vous prions de vous reporter au chapitre "Contrôleur de disquettes", page 207. Le Disk Operating System est compatible au format Microsoft. Les chapitres suivants présentent les différentes routines de l'extramon pour travailler sur les disquettes (gestion de fichiers, backup, etc.). Notons qu'en double densité, le DOS n'utilise que 255 octets par secteur (sur 256 de disponibles).

Initialisation du DOS

Après initialisation de l'EXTRAMON par un RESETC, il est également nécessaire d'initialiser les variables d'EXTRAMON liées aux disques (densité, nombre de disques, nombre de fichiers, etc.). L'ensemble de ces opérations est réalisé par la routine FCBINI.

Pour changer la densité d'un drive (si le contrôleur le permet), il faut modifier la table TABDEN débutant à l'adresse \$621A. Cette table fait 5 octets, chacun étant associé à un lecteur. Pour être en simple densité, il faut mettre \$04 dans l'octet correspondant alors que la valeur \$10 permet de commuter en double densité. Cette table est automatiquement initialisée par FCBINI en fonction du contrôleur présent. Avec un contrôleur double densité la table est entièrement mise à \$10, ce qui est le cas sur le T09. Remarque: il est fortement conseillé de ne pas toucher à la densité du disque virtuel 4, celui-ci est forcément double densité.

Nom	: FCBINI
Code d'entrée	: 02
Paramètres d'entrée	: SECBUF (\$6197) Pointe sur un buffer pour un secteur. 256 ou 128 octets selon la densité. FATPTR (\$6199) Pointe la place libre pour 5 FATS. Il faut DSBLN (166) octets par FAT. Registre Y du 6809E Nombre de disques que l'on veut utiliser (5 au maximum). Accu A du 6809E Nombre maximum de fichiers que l'on veut pouvoir ouvrir en même temps. Registre X du 6809E Pointe la zone libre pour FCBLN (281) x A octets. Réserve la place pour A File Control Block.
Paramètre de sortie	: voir texte
Effet	: Initialisation du DOS

Cache disque

Seul sur les machines T08 et T09+, le cache disque est disponible. Il permet, comme sur d'autres machines concurrentes, d'accélérer notablement les accès disques. Initialement prévu pour optimiser les utilisations des lecteurs QDD, le cache disque peut également être appelé pour des accès aux lecteurs double face, double densité.

Aucun cache n'est défini par défaut. Pour en déclarer l'existence, il est nécessaire de positionner le flag ONOFF (\$62B0) à \$FF et d'adresser une table de descripteurs de caches par le pointeur BLOCS (\$62AE). Chaque descripteur a une longueur de 8 octets divisés de la manière suivante:

KADRESSE	2 octets	Adresse du buffer
KBANQUE	1 octet	Numéro de banque
Reservés	5 octets	

Nous pouvons ainsi déclarer autant de caches que nous voulons. Il suffit d'initialiser les 3 premiers octets de chaque descripteur vers une zone mémoire suffisamment grande pour lire une piste (257*16 en double et 129*16 en simple densité).

Le mot KBANQUE représente un numéro de banque logique compris entre 1 et la valeur écrite dans NBANK (\$618C). Extramon reconnaît la table des descripteurs de cache lorsqu'un 0 est en première position. D'autre part, il suppose que le disque n'est pas retiré du lecteur. Si vous voulez le prévenir d'un risque de changement, il faut positionner le flag RSKCHG (\$6189) à \$FF. Extramon considérera alors ses caches comme incorrects et ira relire le disque si nécessaire. Enfin, EXTRAMON vide ses caches dans les opérations importantes tel que CLOSE, KILL, etc.

Le type de contrôleur est dans l'octet TYPDSK (\$6219). Ainsi le programmeur qui préfère la sécurité à la vitesse peut demander un cache sur QDD et pas sur FDD.

Nom	: INICACHE
Code d'entrée	: 70
Paramètre d'entrée	: Flag ONOFF \$62B0 Registre BLOCS \$62AE Registre RSKCHG \$6189
Paramètre de retour	: Néant
Effet	: Positionne un cache disque après avoir mis à jour les pointeurs et ONOFF=\$FF. Si des opérations sont à réaliser sans cache, il suffit de mettre ONOFF à 0. Par exemple, on peut ainsi interdire les caches en écriture et les autoriser en lecture.

Ouverture d'un fichier

Nom : OPEN

Code d'entrée : 03

On peut ouvrir un fichier selon trois modes:

- Lecture,
- Ecriture,
- Accès direct en lecture/écriture.

Paramètres d'entrée : DK,DRV (\$6049)

Numéro de drive

FILMOD (\$624B)

Type d'accès

M.SQI (\$10) ouvre en input.

M.SQO (\$20) ouvre en output.

MRND (\$40) ouvre en direct.

FILNAM (\$624F-\$6259) Nom du fichier, 11 caractères.

OPTBUF (\$625A-\$6261) Commentaire (écriture seule),
8 octets; s'arrête au 1er \$00.

En écriture, 2 autres flags:

FILTYP (\$624C)

type de fichier.

ASCFLG (\$624D)

\$FF fichier ASCII.

\$00 fichier binaire.

Pour l'ouverture d'un fichier en accès direct, nous vous prions de vous reporter à l'étude de PUTGET (page 272).

Paramètres de retour : FCBNUM (\$6344)

Numéro logique de fichier.

En lecture, 2 autres flags:

FILTYP (\$624C)

Type de fichier.

ASCFLG (\$624D)

\$FF fichier ASCII.

\$00 fichier binaire.

Remarque

: Un numéro de fichier est rendu dans FCBNUM, il permet de distinguer les différents fichiers ouverts en même temps. Pour lire, écrire ou fermer un fichier, il suffit de mettre son numéro dans FCBNUM avant l'appel à la routine concernée. Tout ceci est transparent si l'on n'ouvre qu'un seul fichier.

Lecture d'un caractère

Nom	: INPUT	
Code d'entrée	: 05	
Paramètres d'entrée	: FCBNUM (\$6244)	Numéro logique de fichier
Paramètres de retour	: CARCOU (\$6196)	Le caractère lu
	: EOFLG (\$6178)	Flag de fin de fichier
Remarque	: Si EOFLG est mis, il n'y a pas de caractère valide dans CARCOU.	

Ecriture d'un caractère

Nom	: PRINT	
Code d'entrée	: 04	
Paramètres d'entrée	: FCBNUM (\$6244)	Numéro logique de fichier
	: Accu A du 6809E	Caractère à écrire.

L'accès direct

Nom	: PUTGET	
Code d'entrée	: 07	
Les deux opérations principales d'un enregistrement sont PUT et GET.		
Paramètres d'entrée	: PUTFLG (\$6249)	\$00 pour GET \$FF pour PUT
	: FCBNUM (\$6244)	Numéro logique de fichier
	: Registre X du 6809E	Numéro d'enregistrement désiré
Paramètres de retour	: Pour PUT, le buffer est enregistré	
Paramètres de retour	: Pour GET, le buffer a été lu	

Lors de l'appel de la routine OPEN, il faut préciser la longueur des enregistrements dans RLEN (\$6247-\$6248), ainsi que l'adresse d'un buffer dans BUFFRE (\$62AA-\$62AB). Ce buffer est, bien sûr, de longueur RLEN. Le 1er enregistrement est le numéro 1. Si X est nul, l'enregistrement suivant est pris (séquentiel par défaut).

Remarque : Vous pouvez remplir ou vider le buffer vous-même, ou passer par PRINT et INPUT si vous préférez. Ceux-ci vous préviennent en cas de dépassement du buffer.

Fermeture d'un fichier

Nom	: CLOSE
Code d'entrée	: 06
Paramètre d'entrée	: PCBNUM (\$6244) Numéro logique de fichier.

Lecture du catalogue

Nom	: DIR0 ou DIR1
Codes d'entrées	: 08, 09
Paramètres d'entrée	: DK.DRV (\$6049) Numéro de drive FILNAM (\$624F-\$6259) Filtre 11 caractères.
Paramètres de retour	: NAMSEC (\$618E) Nul, indique fin du catalogue NAMSLT (\$618F-\$6190) Pointe vers 32 octets: octets 0-7 nom octets 8-10 extension octet 11 type de fichier octet 12 flag ASCII binaire octet 13 pointeur dans la FAT octets 14-15 taille du dernier secteur octets 16-23 commentaire sur 8 octets octets 24-31 réservés FACMO (\$6150-\$6151) Taille en octets du fichier

Le registre FILNAM sert de filtre pour le catalogue, des zéros binaires servant de joker. Si tout est à zéro, le catalogue complet est rendu. Le 1er appel se fait par DIR0, par la suite appel à DIR1. A chaque appel, DIR rend un nom de fichier.

Lecture du nom d'une disquette

Nom	: RDVOL
Code d'entrée	: 21
Paramètres d'entrée	: DK.DRV (\$6049) Numéro de drive.
Paramètres de retour	: SECBUF (\$6197-\$6198) Pointe sur le nom (8 octets)

Backup d'une disquette

Trois routines de backup sont disponibles:

- BACKUP0 et BACKUP1 pour les TO8, TO9 et TO9+
- NEW-BACKUP pour les TO8 et TO9+ uniquement.

La différence essentielle du NEW BACKUP par rapport aux deux autres est une meilleure utilisation de l'espace mémoire. Ensuite, les pointeurs DEBZON et FINZON sont sur trois octets, le dernier étant un numéro de banque (entre 1 et NBANK \$618C). DEBZON et FINZON délimitent la zone utilisée dans tous les cas.

Lecteur source différent du lecteur destination

En ce cas, DEBZON et FINZON doivent délimiter une zone mémoire d'au moins la longueur d'une piste, c'est-à-dire 2 ou 4 Ko, selon la densité du drive, sinon une erreur est générée.

Lecteur source identique au lecteur destination

Ce BACKUP utilise toute la mémoire, de la banque 1 à la banque NBANK (\$618C). Vous pouvez toujours modifier NBANK pour préserver votre mémoire, le BACKUP n'en sera que plus long. Le contrôle vous est rendu lorsqu'un changement de disquette est nécessaire: à vous d'afficher le message nécessaire et de continuer le BACKUP par un appel à BACKUP1. Le flag SWPFLG vous indique une demande de SWAP. Si SWPFLG est nul, c'est que l'opération est terminée.

Fiche des routines

Pour TO8, TO9 et TO9+:

Noms	: BACKUP0, BACKUP1 -
Nombres points d'entrée	: 10, 11
Pour TO8 et TO9+ uniquement :	
Nom	: NEW-BACKUP
Code d'entrée	: 71
Paramètres d'entrée	: DK.DRV (\$6049) Drive destination. Accu A du 6809E Drive source. DEBZON (\$616B-\$616C) Zone mémoire de travail. FINZON (\$616E-\$616F) Fin de cette zone.
Paramètres de retour	: SWPFLG (\$619D) Flag : doit-on swapper ?

Copie d'un fichier

La taille minimum du buffer est de 20 octets. Mais une taille assez grande est presque indispensable dans le cas de copie avec SWAP.

Lors de la copie, le commentaire du fichier origine est recopié si le nouveau commentaire commence par un 0. La date du fichier origine sera recopiée dans le catalogue du nouveau fichier (TO8 et TO9+ uniquement).

Comme pour BACKUP, 2 cas se présentent. Le cas d'une copie nécessitant un swap et l'inverse. Si les registres X et Y du 6809E sont égaux, EXTRAMON suppose que l'utilisateur veut copier son fichier dans le même lecteur, mais sur une autre disquette et avec le même nom. Dans ce cas, si la zone tampon n'est pas assez longue pour contenir le fichier, COPY rend la main avec SWPFLG différent de 0. C'est alors au programme appelant d'afficher un message du style "PLEASE INSERT DESTINATION etc." et lorsque l'utilisateur a changé sa disquette, il faut rappeler COPY1 qui continue le travail. C'est fini pour le programme appelant lorsque SWPFLG est nul.

Si X est différent de Y, il n'y a aucun problème et rien d'autre à faire. Attention, il faut avoir réservé au moins 2 FCBs (voir FCBINT) pour pouvoir faire COPY.

Le point d'entrée NEW-COPY (disponible uniquement sur TO8 et TO9+) est très proche de COPY0, la différence principale étant une meilleure gestion mémoire. Les pointeurs DEBZON et FINZON sont alors sur trois octets, le dernier étant un numéro de banque (entre 1 et NBANK \$618C). Dans tous les cas, ces deux pointeurs délimitent la zone utilisée.

Enfin pour une copie mono lecteur, la suite se fait par appel de COPY1.

Noms	: COPY0, COPY1
Codes d'entrée	: 12, 13
Paramètres d'entrée	: Registre X du 6809E Pointe fichier source. Registre Y du 6809E Pointe fichier destination.

Les registres X et Y pointent sur une zone de 20 octets:

octets 0-7	nom
octets 8-10	extension
octets 11-18	commentaire
octet 19	numéro de drive.

DEBZON (\$616B-\$616C) Zone mémoire de travail.

FINZON (\$616E-\$616F) Fin de cette zone.

Paramètres de retour	: SWPFLG (\$619D) Flag: doit-on swapper?
----------------------	--

Destruction d'un fichier

Noms	: KILL
Code d'entrée	: 14
Paramètres d'entrée	: DK.DRV (\$6049) Numéro de drive. FILNAM (\$624F-\$6259) Nom du fichier. 11 caractères. 8 octets pour le nom. 3 octets pour l'extension
Paramètres de retour	: Sauf erreur, votre fichier est perdu! Entre nous, vous l'avez bien cherché.

Changement de nom d'un fichier

Nom	: NAMR
Code d'entrée	: 15
Paramètres d'entrée	: Registre X du 6809E Pointe fichier ancien nom. Registre Y du 6809E Pointe fichier prochain nom.

X et registre Y pointent sur une zone de 20 octets:

octets 0-7	nom
octets 8-10	extension
octets 11-18	commentaire
octet 19	numéro de drive

EXTRAMON vérifie que le nouveau nom n'est pas déjà présent sur la disquette. Lors du changement de nom, le commentaire du fichier est conservé si le nouveau commentaire commence par 0. La date du fichier est conservée (TO8 et TO9+ uniquement).

Initialisation d'une disquette

Nom	: DSKINI
Code d'entrée	: 17
Paramètres d'entrée	: DK.DRV (\$6049) Numéro de disque FILNAM (\$624F-\$6256) Nom de volume (8 octets) DK.NUM (\$604D) L'entrelacement désiré (7 est bien) VERFLG (\$618D) \$00 pas de vérification \$80 vérification (beaucoup plus lent)

Place libre sur une disquette

Nom	: DSKF
Code d'entrée	: 16
Paramètres d'entrée	: DK.DRV (\$6049) Numéro de drive
Paramètres de retour	: FACMO (\$6150-\$6151) La place libre en Koctets

Taille d'un fichier

Nom	: LOF
Code d'entrée	: 18
Paramètres d'entrée	: RCBNUM (\$6244) Numéro logique de fichier
Paramètres de retour	: FAC (\$614E) Résultat en entier 16 bits (FACMO(\$6150)) ou en réel 4 selon la taille, pour le savoir, il faut tester VALTYP (2 ou 4).

Dans le cas des fichiers à accès direct, c'est le nombre d'enregistrements du fichier qui est rendu. En séquentiel, c'est le nombre de secteurs du fichier.

Formule de calcul pour l'accès direct :

$$\text{LOF} := (\text{NBoct} * \text{NBsecteur} - \text{Inutiles}) / \text{RLLEN}$$

Avec: NBoct = Nombre d'octets par secteur
 NBsecteur = Nombre de secteurs du fichier
 Inutiles = Nombre d'octets inutiles du dernier secteur
 RLEN = Taille d'un enregistrement

Attention: Sur le TO9, en double densité EXTRAMON prend NBoct à 256 octets au lieu de 255, d'où des petites erreurs de calcul... Ce problème n'existe plus sur TO8 et TO9+.

Numéro d'enregistrement courant

Nom	: LOC
Code d'entrée	: 19
Paramètres d'entrée	: RCB (\$6244) Numéro logique de fichier
Paramètres de retour	: FAC (\$6150-\$6151) Résultat en entier 16 bits.
Effet	: LOC rend le numéro de l'enregistrement courant sur un fichier à accès direct. Sur un fichier à accès séquentiel, c'est le numéro du secteur courant qui est rendu.

Exemple d'utilisation

Le programme ci-dessous représente une application simple du DOS en assembleur. Son but est d'afficher à l'écran le nom d'une disquette :

	ORG	\$A000	
EXTRA	EQU	\$EC0C	
PUTC	EQU	\$E803	
SECBUF	EQU	\$6197	
DK.DRV	EQU	\$6049	
RDVOL	EQU	21	
	LDX	#NBUF	Initialisation du pointeur.
	STX	SECBUF	SECBUF sur NBUF.
	LDB	#01	Initialisation d'EXTRAMON.
	JSR	EXTRA	-
	CLRA		
	STA	DK.DRV	Numéro de drive 0
	LDB	#RDVOL	Recherche du nom de disque.
	JSR	EXTRA	-
	LDX	#NOM	Affichage.
REC	LDB	,X+	-
	JSR	PUTC	-
	CMPX	#FBUF	
	BNE	REC	-
	SWI		
NOM	FCC	/NOM DU DISQUE/	
NBUF	RMB	8	
FBUF	EQU	*	
	END		

6. L'éditeur

Cette routine permet d'utiliser un éditeur ayant toutes les fonctionnalités de l'éditeur BASIC (insertion, effacement, déplacement curseur, etc.). Il fonctionne aussi bien en mode TO7-70 qu'en mode 80 colonnes. L'édition se fait dans la fenêtre courante, nous vous conseillons de vous reporter à l'étude de la routine PULC du moniteur et sur ses séquences d'échappements US pour fixer cette fenêtre (page 175)

En entrée l'appelant fournit un buffer, en sortie il reçoit la ligne lue. L'éditeur teste (entre autres) l'octet IWTF LG. Dans le cas où ce dernier est différent de zéro, on sort de l'éditeur avec un tampon vide. Ceci permet quelques fantaisies comme par exemple reprendre la main sur une touche fonction ! Pour ce faire, le registre IWTF LG peut être modifié dans une routine d'interruption ou dans l'indirection de GEIC du moniteur.

Il est également possible de demander à l'éditeur de prendre un texte pointé par DEFTXT à la place de l'entrée clavier, le texte devant se terminer par \$00. Pour ne pas avoir de texte par défaut, il est recommandé de pointer DEFTXT sur un \$00.

A titre d'exemple, citons l'instruction AUTO du BASIC qui fonctionne selon ce principe. En entrant dans l'éditeur, DEFTXT pointe sur le numéro de ligne à afficher; ainsi ce numéro de ligne est écrit à l'écran comme si l'utilisateur l'avait frappé.

Comme nous le disions quelques lignes auparavant, grâce à l'indirection de GEIC, l'application peut dériver DEFTXT sur un texte lors d'un appui sur une touche fonction (F1 déclenche un RUN, F2 un LIST, etc.). En fait, il faut que la routine indirectée de GEIC rende le 1er caractère "L" et fasse pointer DEFTXT vers 1ST,\$00. Si par exemple, il y a un CR dans la ligne, elle est automatiquement validée par l'éditeur.

Nom	: EDIT
Code d'entrée	: 22
Paramètres d'entrée	: Registre X du 6809E Adresse du tampon Registre Y du 6809E Fin du tampon (dernier octet utile) DEPTXT (\$61E0-\$61E1) Pointe sur un texte se substituant à l'entrée clavier. IWTFLG (\$62A9) Sémaphore de sortie rapide
Paramètre de sortie	: Néant
Effet	: Le tampon est rempli lors de la frappe de la touche ENTRÉE. La fin de la ligne se reconnaît par un \$00. Les accents sont codés avec les séquences SS2. Les caractères non reconnus par l'éditeur sont effacés à l'écran. Si l'utilisateur frappe CNT-C, EDIT rend la main à l'application avec un tampon vide.
Remarque	: Il ne faut pas fournir de tampon plus petit que 3 caractères.

7. L'interpréteur musical

La routine PLAY est à l'extramonteur ce que l'instruction PLAY est au BASIC. C'est dire sa facilité d'utilisation ! Après avoir précisé dans l'accumulateur A la longueur de la chaîne à jouer et pointé par le registre Y l'adresse où elle se trouve, il suffit d'appeler la routine PLAY pour que vos oreilles mélomanes jouissent de vos talents de compositeur. La syntaxe de la chaîne à interpréter, son contenu, ses paramètres ainsi que ses valeurs par défaut sont rigoureusement les mêmes que pour l'instruction PLAY du BASIC (Octave, durée, tempo, etc.)

```

Nom          : PLAY
Code d'entrée : 23
Paramètres d'entrée : Registre Y du 6809 E
                  : Accumulateur A du 6809 B
Paramètres de retour : Néant
Effet        : Exécute la mélodie désignée par Y
    
```

Exemple

```

* Balayage de la gamme de DO octave 3 *
* DO octave 3 (dernière note est ronde)
    
```

```

# TITLE      EXMUSIC
# ORG        #A000
EXTRA EQU    $E000
RESETW EQU   01
PLAY EQU     23

FLAG EQU     *      RESET GENERAL
LDA #RESETW
JSE EXTRA

FLAG1 EQU    *      prog etudia
LDA #39      longueur chaîne
LDY #MUSIC
LDR #PLAY
JSE EXTRA
SWI

MUSIC FCB    /03DOREMIFASOLASI/
FCB    /04DOREMIFASOLASI/
FCB    /060506/

END
    
```

8. Les messages d'erreur en anglais

Les messages d'erreur générés sous BASIC représentent une suite de chaînes de caractères qui peuvent être appelés par l'extramonitor. La procédure est la suivante:

- Tout d'abord on implante dans l'accumulateur A le numéro du message d'erreur dont on désire l'affichage (par exemple 02 pour "Syntax Error").
- Ensuite il faut pointer par le registre d'index X un buffer où sera rangée la chaîne de caractères correspondante.
- Enfin vous appelez la routine ERRMSG qui plantera le message dans le buffer pointé.

Notez que la routine ne se charge pas de l'affichage. Ainsi, dans l'exemple proposé ci-dessous nous utilisons la routine PUTC ("Affichage des caractères alphanumériques, page 163) dans une boucle, afin de transférer sur l'écran le contenu du buffer.

```

Nom          : ERRMSG
Code d'entrée : 20
Paramètres d'entrée : Accumulateur A du 6809E
                  : Registre X du 6809E
Paramètres de retour : Registre X du 6809E
Effet        : Transfert dans un buffer pointé par X, le message d'erreur
                  répété par A.
Exemple      : Le programme ci-dessous vous offrira un spectacle que
                  j'espère original pour vous, puisqu'il affiche une
                  trentaine de messages d'erreurs simultanément !

* Affichage des 28 derniers messages
* d'erreurs du basic (du no 50 au 78)

          TITLE    EXMESS
          ORG      $A000
EXTRA EQU $EC0C
PUTC EQU $E803
RESETW EQU 01
ERRMSG EQU 20

FLAG EQU *      RESET GENERAL
LDB #RESETW
JSR EXTRA

```

```

FLAG1 EQU *          prog etudie
      LDA #50         message numero 50
AUTMES LDX #18000     implant tampon
      LDB #ERRMSG     routine numero 20
      JSR EXTRA
AFICHE LDB ,X+        affichage du
      JSR PUTC        tampon par PUTC
      CMPB #00        test fin de chaine
      BNE AFICHE
      INCA
      CMPA #79        test dernier mess
      BNE AUTMES     message suivant
      SWI
      END

```

Note : Les numéros des messages d'erreurs BASIC sont répertoriés dans les guides accompagnant les unités centrales.

Le DOS iconique

Généralités

Le DOS Iconique, appelé sur les menus "Exploitation de fichiers", est situé sur la banque 3 du slot 0 des T08, T09 et T09+. Il utilise pour réaliser ses différentes manipulations de fichiers, des routines du moniteur et de l'extramoniteur. Ainsi, globalement, ses variables se situent dans les pages \$60, \$61 et \$62, ce qui implique que toute application utilisant le DOS Iconique doit laisser ces 3 pages libres.

Comme toutes routines de l'extramoniteur, le début du programme sera consacré aux initialisations des sous-ensembles concernés:

- Unités de disquettes
- Fenêtre graphique (nulle par défaut)
- Curseur graphique qui définit la position des coins gauches des tableaux générés par le DOS Iconique.
- Mise à jour du registre COULEUR (\$619F)
- Le registre CHDRAW (\$6041) doit être à 0.

Le DOS Iconique ne sauvegarde pas l'écran avant d'afficher ses fenêtres, il faudra donc prévoir un traitement adéquat dans vos programmes personnels.

Trois routines sont utilisables par une application externe:

- La sélection de fichiers,
- La saisie d'un nom de fichier,
- La sélection du lecteur courant.

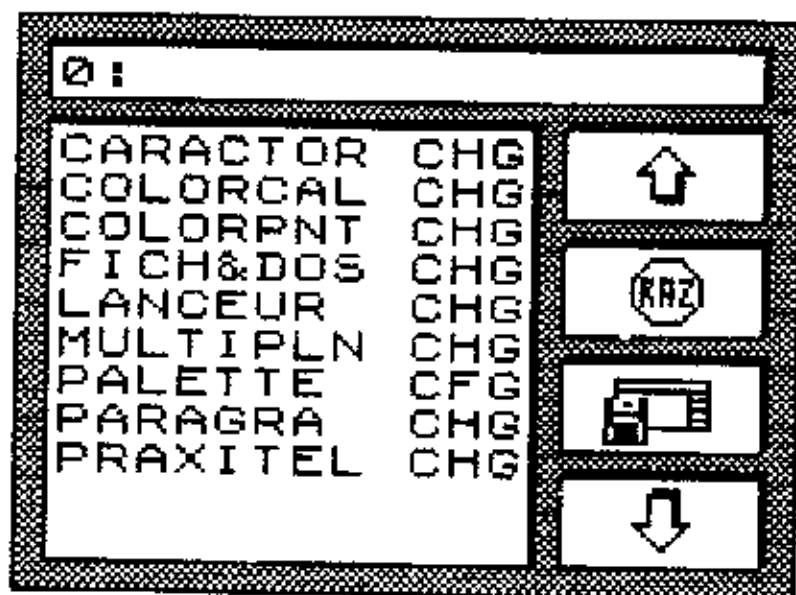
Ainsi, l'utilisateur peut à l'aide du crayon optique ou de la souris (si elle est en action), sélectionner un fichier, relire le catalogue et se déplacer à l'intérieur. Toutes les commandes light pen ou souris sont doublées au clavier par les touches RAZ, ENTREE, les chiffres de 0 à 4 et les flèches verticales.

Pour réaliser ces tâches, un buffer pointé par le registre d'index X doit être alloué, sa longueur étant définie par le registre Y. Si ce buffer est trop petit, une erreur "Out of Memory" sera générée.

Sélection de fichiers

La routine DIRR fournit un catalogue réduit (nom + extension), classé alphabétiquement, de la disquette placée dans le lecteur courant. Au préalable, les registres DK.DRV et FILNAM indiqueront respectivement le numéro de lecteur concerné et le filtre de sélection des noms de fichiers (0 étant le caractère transparent). En sortie, le registre B retourne le numéro de l'erreur. Si son contenu vaut 0, aucune erreur n'est détectée et le nom est inscrit dans le registre FILNAM.

Un buffer de 1500 octets est nécessaire pour accomplir cette routine, sachant que la fenêtre affichée à l'écran a une taille de 20 x 15 caractères.



Saisie d'un nom de fichier

La routine SAISIE permet la frappe d'un nom de fichier complet constitué du nom, de l'extension et du commentaire. Le registre FILNAM contient le nom par défaut proposé à l'utilisateur suivi de l'extension et du commentaire (8+3+8 caractères).






La saisie est validée par le pointage de OK dans le tableau affiché ou par la frappe de la touche ENTREE. La touche RAZ ou le pointage de "Annuler" interrompt la saisie.

Un buffer de 100 octets est nécessaire pour cette routine. Le tableau affiché occupe 24 x 11 caractères.

Nom	PARAGRA	CHG
Commentaire		
<div>OK <input type="checkbox"/> Annuler <input type="checkbox"/></div>		

Sélection du lecteur courant

La routine SELDEV permet de sélectionner le lecteur courant. Le registre DK.DRV contient le numéro du lecteur à mettre en évidence. Un buffer de 100 octets est nécessaire. La taille de la fenêtre affichée est de 24 × 11 caractères.

0	1	2	3	4
				

Appel au DOS Iconique

L'appel de l'une des routines décrites ci-dessus se fait via la routine COMS du moniteur. Pour de plus amples détails, veuillez vous reporter au chapitre "Commulation des mémoires ROM", page 222.

Exemple:

	LDX	#\$8000	Début du buffer
	LDY	#1700	Longueur de la zone (1700 octets)
	LDA	#03	Appel du slot 0, banque 3
	LDU	#ROUTINE	Point d'entrée de la routine ex \$3FC1 pour DIRR
	JSR	COMS	
	TSTR		
	BNE	ERROR	
OK	EQU		*

Il conviendra d'ajouter au début du programme les initialisations des registres concernés par la routine appelée. Vous trouverez ci-dessous un résumé des equates du DOS Iconique.

DK.DRV	EQU	\$6049	Numéro de lecteur courant
SECBUF	EQU	\$6197	Pointer de buffer disque
FATPTR	EQU	\$6199	Pointer de buffer FAT
FILNAM	EQU	\$624F	Nom de fichier courant
CHDRAW	EQU	\$6041	Caractère graphique
COULEUR	EQU	\$619F	Couleur courante
EXTRA	EQU	\$FC0C	Point d'entrée d'extramon
COMS	EQU	\$EC03	Point d'entrée de commutation de slot
DIRR	EQU	\$3FC1	Point d'entrée du catalogue réduit
SAISIE	EQU	\$3FC4	Point d'entrée de SAISIE
SELDEV	EQU	\$3FC7	Point d'entrée de SELDEV

10. Informations complémentaires

Extramon sous BASIC 512

Le BASIC 512 disponible sur les T08 et T09+, initialise automatiquement l'extramoteur. En conséquence, vos programmes personnels appelant extramon à partir du BASIC 512 n'ont pas à réaliser ce traitement. Si vous désirez récupérer les erreurs éventuelles, il faut rediriger le vecteur de rattrapage d'erreur ZPERR (\$6185) de la manière suivante:

```
SAVSTK EQU $6175
LDS     SAVSTK
PULS    A,DP,X,Y,U,PC
```

Dans le cas contraire, c'est le message d'erreur BASIC qui sera affiché.

Les numéros de fonctions ou routines d'extramon

RESETC	EQU	0	reset à froid
RESETW	EQU	1	reset à chaud
FCBINI	EQU	2	init d'un FCB
OPEN	EQU	3	ouverture
PRINT	EQU	4	output
INPUT	EQU	5	input
CLOSE	EQU	6	close
PUTGET	EQU	7	accès direct
DIR0	EQU	8	catalogue
DTR1	EQU	9	catalogue suite
BACKUP0	EQU	10	backup
BACKUP1	EQU	11	backup suite
COPY0	EQU	12	copie
COPY1	EQU	13	copie suite
KILL	EQU	14	destruction d'un fichier
NAME	EQU	15	renomme un fichier
DSKF	EQU	16	calcul place libre
DSKINI	EQU	17	initialise la disquette
LOF	EQU	18	taille d'un fichier
LOC	EQU	19	adresse écriture dans fichier

FRRMSG	EQU	20	rend erreur en clair
RDVOL	EQU	21	va lire le nom du disque
EDIT	EQU	22	l'éditeur plein écran
PLAY	EQU	23	l'interpréteur musical
CIRCLE	EQU	24	dessin du cercle ou ellipse
PSBTXY	EQU	25	écrit un point
LINE	EQU	26	trace une ligne
BOX	EQU	27	trace un rectangle
CHOIX	EQU	28	initialisation mode graphique
PAINT	EQU	29	remplissage d'une surface
MIG	EQU	30	l'interpréteur graphique
TRACE	EQU	31	tracé de tortue
ANIME	EQU	32	liberté de la tortue
SHOW	EQU	33	allumage de la tortue
HEAD	EQU	34	direction de la tortue
ROT	EQU	35	rotation de la tortue
ZOOM	EQU	36	taille de la tortue
FWD	EQU	37	avance de la tortue
MOVE	EQU	38	déplacement de la tortue
INTORTUE	EQU	39	initialise une tortue
CMPTORTUE	EQU	40	compile une forme de tortue
SGN	EQU	41	signe
INT	EQU	42	entier
ABS	EQU	43	valeur absolue
SQR	EQU	44	racine carrée
LOG	EQU	45	logarithme népérien
EXP	EQU	46	exponentielle
COS	EQU	47	cosinus
SIN	EQU	48	sinus
TAN	EQU	49	tangente
FRCTYP	EQU	50	force le type
FIXER	EQU	51	troncature
FRND	EQU	52	valeur aléatoire
NEGGO	EQU	53	négation
ADDGO	EQU	54	addition
SUBGO	EQU	55	soustraction
MULTGO	EQU	56	multiplication
DIVGO	EQU	57	division réelle
EXPGO	EQU	58	exponentiation
IMODO	EQU	59	reste de la division entière
IDIVO	EQU	60	division entière
MOVFM	EQU	62	FAC := mémoire
MOVMF	EQU	63	mémoire := FAC
MOVAF	EQU	64	ARG := FAC
FIN	EQU	65	conversion ASCII == > binaire
PUFOUT	EQU	66	C.binaire == > ASCII décimal
HOFOUT	EQU	67	C.binaire == > ASCII hexa/octal

***POUR T08 ET T09+ UNIQUEMENT:**

ATN	EQU	68	arctangente
CODE	EQU	69	code et décode image
INICACHE	EQU	70	cache disque
NEWBACKUP	EQU	71	backup
NEWCOPY	EQU	72	copie

Les equates d'extramem

******* MODES D'OUVERTURE POUR LA ROUTINE OPEN *******

M.SQI	EQU	\$10	ouverture en input séquentiel
M.SQO	EQU	\$20	ouverture en output séquentiel
M.RND	EQU	\$40	ouverture en direct (I/O)

******* LES OBJETS TORTUES *******

TX	EQU	6	position de la tortue en X
TY	EQU	9	position de la tortue en Y
TROT	EQU	12	rotation de la tortue
TTAI	EQU	13	taille de la tortue
TDIR	EQU	14	direction de la tortue
TFORME	EQU	16	forme de la tortue

******* L'ACCUMULATEUR FLOTTANT *******

DBLFLG	EQU	\$6103	flag de double précision
VALTYP	EQU	\$6105	indicateur de type
FAC	EQU	\$614E	accumulateur
FACEXP	EQU	\$614F	exposant
FACHO	EQU	\$614F	octet fort de la mantisse
FACMO	EQU	\$6150	octet moyen de la mantisse
FACLO	EQU	\$6151	octet faible de la mantisse
DFACHO	EQU	\$6152	4 octets de plus pour double précision
DFACMH	EQU	\$6153	
DFACML	EQU	\$6154	
DFACLO	EQU	\$6155	
FACSGN	EQU	\$6156	signe de FAC (0 ou -1) quand non codé

*****LES ARGUMENTS FLOTTANTS (NON CODES)*****

ARGEXP	EQU	\$6159	
ARGHO	EQU	\$615A	
ARGMO	EQU	\$615B	
ARGLO	EQU	\$615C	
DARGHO	EQU	\$615D	4 octets de plus pour double précision
DARGMH	EQU	\$615E	
DARGML	EQU	\$615F	
DARGLO	EQU	\$6160	
ARGSGN	EQU	\$6161	
*			
DEBZON	EQU	\$616B	Début et fin de zone tampon pour
FINZON	EQU	\$616E	EXTRAMON
EOFFLG	EQU	\$6178	flag fin de fichier zéro - non fini non zéro - fini

***** TEMPORAIRES POUR PRINT USING*****

DPWID	EQU	\$617A	
FLDWID	EQU	\$617B	
PUMASK	EQU	\$617C	
ZPERR	EQU	\$6185	rattrapage d'erreur d'EXTRAMON BASIC l'initialise à ERROR dans CLEAR
RSKCHG	EQU	\$6189	Flag de risque de changement de disque
NBANK	EQU	\$618C	nombre de banques accessibles

***** VARIABLES LIEES AUX DISQUES *****

VERFLG	EQU	\$618D	pour VERIFY ON ou OFF
NAMSEC	EQU	\$618E	le secteur du catalogue contenant le nom recherché.
NAMSLT	EQU	\$618F	un pointeur vers le slot dans le secteur (voir NAMSEC), ou zéro si le nom n'a pas été trouvé.

CARCOU	EQU	\$6196	Le caractère que DECHRI vient de lire. SECBUF et FATPTR doivent être positionnés avant le 1er appel au DOS sous peine d'erreur.
SECBUF	EQU	\$6197	Un pointeur vers le buffer de lecture d'un secteur
FATPTR	EQU	\$6199	pointeur vers la zone où l'utilisateur veut ranger les FATs (DSRs)
DSBLEN	EQU	6+2*80	
SWPFLG	EQU	\$619D	un flag pour dire que l'on veut un swap disquette.

***** VARIABLES GLOBALES NECESSAIRES AU GRAPHIQUE *****

COULEUR	EQU	\$619F	
TRATYP	EQU	\$61A0	type de tracé 0 pour normal 1 normal sans couleur 2 en ou exclusif
XXXX	EQU	\$61A1	
YYYY	EQU	\$61A3	le curseur graphique.
XL	EQU	\$61A5	La fenêtre graphique
YB	EQU	\$61A7	
XR	EQU	\$61A9	de XLeft, YBottom
YT	EQU	\$61AB	à XRight, YTop.

***** TEMPORAIRE POUR L'EDITEUR *****

DEFTXT	EQU	\$61E0	pointeur vers du texte à afficher
--------	-----	--------	-----------------------------------

***** VARIABLES LOCALES AU TRACE D'ELLIPSES *****

FILFLG	EQU	\$61EF	doit-on remplir l'ellipse ou le rectangle...
AXEV	EQU	\$61F0	axe vertical.
AXEH	EQU	\$61F1	axe horizontal.
CAMFLG	EQU	\$61F2	camembert ?
ALPHA1	EQU	\$61F3	
ALPHA2	EQU	\$61F7	

***** VARIABLES LOCALES AU CODAGE ET AU DÉCODAGE

XOCOD	EQU	\$61D6	extrémités du rectangle
YOCOD	EQU	\$61D7	
XICOD	EQU	\$61D8	
YICOD	EQU	\$61D9	

***** DIVERS de \$6200 à \$62FF *****

TYPDSK	EQU	\$6219	Type de contrôleur
TABDEN	EQU	\$621A	table des densités par disque
MAXMOD	EQU	\$6223	
FCBNUM	EQU	\$6244	numéro du FCB courant
RLEN	EQU	\$6247	longueur de l'enregistrement du fichier à accès direct.
PUTFLG	EQU	\$6249	pour distinguer PUT de GET !
FILMOD	EQU	\$624B	mode du fichier (OPEN).
FILTYP	EQU	\$624C	type de fichier. 0 = BASIC program. 1 = BASIC data file. 2 = machine language file.
ASCFLG	EQU	\$624D	Flag ASCII. 0 = fichier non ASCII. FF = fichier ASCII.
FILNAM	EQU	\$624F	buffer nom de fichier.
FILEXT	EQU	\$6257	buffer extension nom de fichier
OPTRBUF	EQU	\$625A	buffer de descriptions des options.
MACP	EQU	\$627D	le pointeur vers le motif de peinture.
WTTH	EQU	\$6288	flag avec ou sans couleur.
IWTFLG	EQU	\$62A9	pour l'éditeur, lorsque ce flag devient ⇔ de 0, on sort sans réfléchir !!!
BUFFRE	EQU	\$62AA	pour l'OPEN en accès direct. Pointe vers un buffer (de longueur RLEN) où l'appelant retrouvera l'enregistrement demandé.
ONOFF	EQU	\$62B0	Flag avec ou sans disque
BLOCS	EQU	\$62AE	Pointe sur une table de descripteur de blocs