



## Le Sharp PC G850V

Sortie en 1995, le G850V est le dernier de la lignée des ordinateurs de poche façon 80's (avec le VS, sorti en 2009, est quasi identique), il reprend le concept ordi/caltoche amorcé par le 1401, repris ensuite par les séries 14xx, E2xx, E5xx, E8xx.

Il se présente sous la forme d'un boîtier anthracite en résine protégé par un couvercle amovible basculable sur la face avant ou la face arrière. A noter que celui-ci permet accès au port RS232 par une découpe judicieuse.

Sur la face arrière, par une trappe ouvrable à la main donne accès aux 4 piles type AAA offrant une très bonne autonomie au PC (plusieurs mois) mais exigeant une épaisseur conséquente du boîtier (198x102x27mm avec



boîtier de protection).

Basé sur un processeur Z80 à 8Mhz il est fourni avec 96Ko de ROM (selon le manuel, mais j'ai quelques doutes) et 32Ko de RAM (30179 octets aucun doute là).

Outre le port du bus système accessible par l'ouverture d'un cache sur son côté droit, on ne dispose ici d'aucun compartiment d'extension, notamment les extensions de mémoire, au vu des fonctionnalités de la machine c'est une franche déception.

De part son processeur et sa philosophie générale, il est le fils légitime des PC1500 et PC1600 (comme toute la gamme G8xx d'ailleurs), Cependant son Basic, son port RS232 (à gauche), l'absence de TIME ... sont hérités la série 12, 13, 14 et suivantes (processeur SC61860 et compatible), une sorte de fusion des deux lignées en somme, un aboutissement.

La coque plastique est l'une des évolutions (si j'ose dire !) les plus visibles, exit donc le beau

boîtier sensuel en aluminium brossé des versions 80, on est ici dans les années 90 et la réduction des coûts a frappée (TI était précurseur). Le plastique est cependant de bonne qualité (d'un niveau HP48 je dirais) et plutôt bien ajusté sans craquements intempestifs (bref ce n'est pas un FX702P quoi !!).

**Le clavier** est d'un touché agréable et permet une frappe rapide à deux doigts. Segmenté en deux, sa partie droite est dédiée aux fonctions mathématiques directes et sa partie gauche à l'alphabétique. La saisie de programme ou de texte est facilitée par l'apparition d'une touche [INS]DEL permettant la bascule entre deux modes de saisie. Soit l'écrasement de caractère, soit l'insertion. On s'embête nettement moins que les séquences interminable de [2<sup>nd</sup>][INS] et [2<sup>nd</sup>][DEL] bien connue des possesseurs des générations originelles.

Une touche [BACKSPACE] participant aussi au confort, que l'on trouvait déjà sur le 1600, permet, elle, l'annulation du dernier caractère frappé. Notez que tant les flèches que [BACKSPACE] et [DEL] sont à répétition.

La touche [SHIFT] fonctionne différemment des autres pockets série 12, 13, 14, 15 et 16 et demande son maintien en pression pour l'accès à la fonction (comme sur un ordinateur contemporain en fait), à noter que la touche, côté pavé numérique, [2<sup>nd</sup>] permet le SHIFTAGE selon la tradition SHARP avec un appui séquentiel.

Les majuscules sont accessibles par [CAPS] en utilisation bascule. Le shiftage des touches alphabétiques, donnant lui, accès à quelques mot clé BASIC (inutile pour moi car le Sharp gère les abréviations) ou caractère spéciaux.

La touche [BASIC] permet soit d'avoir accès au mode BASIC, avec toujours sa déclinaison PRO, dédié à la programmation, et RUN permettant lui l'exécution de programme ou de commande directe, de calcul d'expressions ou de calcul façon caltoche's 80.

La touche [TEXT] elle appelle l'éditeur de texte intégré qui permet de saisir des sources en C, en CASL, en assembleur ou même en mémo (tubes et autre pense bête illicite) et même en basic (quoique dans ce cas le mode Basic vérifiant la syntaxe sur validation de la ligne est plus approprié), avec un impératif malheureux ... les numéros de lignes sont obligatoires !



Les majuscules sont gérées, ainsi que plusieurs jeux de caractères japonais accessibles par l'appui sur la touche idoine (c'est avant tout une machine dédiée au marché japonais avec une cible étudiant en informatique).

A noter aussi le bouton RESET accessible en façade et de diamètre suffisant pour utiliser la pointe d'un stylo.

Le shiftage de la touche [ENTER] donnant accès au mode bien connu des utilisateurs SHARP permettant l'impression sur le CE126P (PRINT=LPRINT ou PRINT=PRINT).

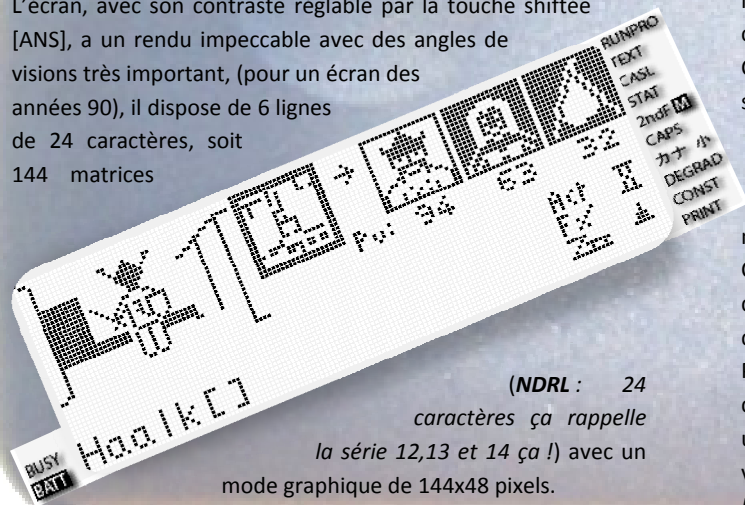
Le mode DEF des anciennes générations, pourtant bien utile, est lui inexistant, il sera pallié en parti par la gestion de fichier intégrée ne prenant pas en charge le MERGE ni le CHAIN malheureusement.

Pas de touches de fonctions (une ligne de touche contre deux lignes d'affichage on va dire).

Pour le jeu de caractères géré il est codé en ASCII sur 8 bits (256 caractères) mais ne dispose d'aucun caractère accentué, pas étonnant vu sa cible mais quand même regrettable.

### Un format d'écran inédit

L'écran, avec son contraste réglable par la touche shiftée [ANS], a un rendu impeccable avec des angles de visions très important, (pour un écran des années 90), il dispose de 6 lignes de 24 caractères, soit 144 matrices



(NDRL : 24 caractères ça rappelle la série 12,13 et 14 ça !) avec un mode graphique de 144x48 pixels.

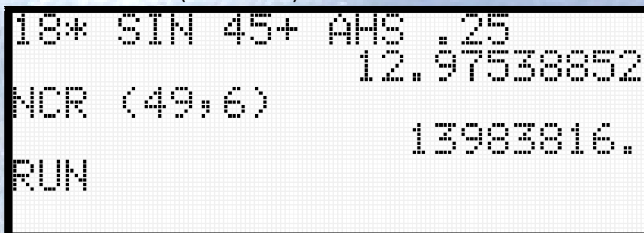
Imposant par rapport au 80's .... Mais avec la pléthore de langages il eu été bon de voir plus grand avec un écran de 6, ou même 8 lignes et 40 caractères (le E500 a bien 4x40=160 caractères lui !! avec un clavier alpha plus conséquent). On se sent très à l'étroit pour une programmation en C ou même en assembleur.

L'écran possède sur sa gauche l'indicateur [BUSY] et [BATT] annonçant un manque d'énergie. Sur la partie droite ce n'est pas

moins de 15 notifications différentes annonçant les diverses états du PC

### La caltoche

Le mode calculatrice n'est pas spécifique, contrairement à la série 14 et même à la concurrence (Casio pour être clair), on l'utilise en basic (mode RUN).



La partie droite de la calculatrice agit façon inversée si une valeur numérique est saisie la fonction est appliquée à cette valeur (comme une calculatrice des années 80 en fait). Si aucune valeur n'est saisie, le nom de la fonction apparait sur la ligne de commande pour l'édition d'une expression à calculer avec toutes les possibilités de rappel de changement, bref le traditionnel chez Sharp.

C'est moins spécialisé que 2 modes séparés mais par expérience le mode calculatrice des Sharp PC 14xx n'est pas forcément très exploité, la possibilité de pouvoir réédité son calcul (ou du moins le vérifier) étant nettement privilégié. Ça devient une évidence avec les calculateurs des années 90, 2000 et certainement 2010 au vu de l'évolution actuelle.

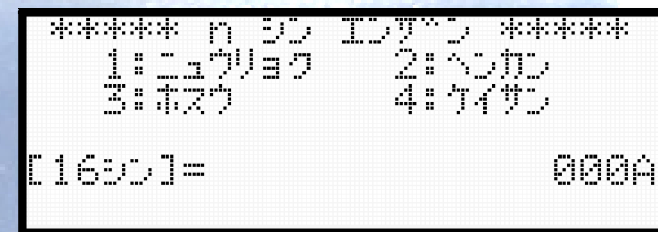
Plusieurs originalités sur la partie calculatrice, d'abord la touche [CONST] permettant de mémoriser une opération de +, -, \* ou / par un nombre sur validation par [ENTER].

Un exemple simple si je frappe [CLS]\*(7/2)[CONST], chaque nombre saisie suivi d'une validation se verra multiplié par 3.5. On sortira de ce mode par [2<sup>nd</sup>][CA], un indicateur [CONST] apparait dans ce mode sur la partie droite de l'écran.

Ensuite [MDF] fonctionnant en parallèle de [DIGIT] (une sorte de fix) qui permet de tronquer aux décimales non affichées, intéressant par exemple lorsque les calculs intermédiaire sont à tronquer à la x<sup>ème</sup> décimale.

variables avec tous les calculs traditionnels mais les menus sont en japonais il faut donc chercher un peu.

Le mode [Base-n], lui aussi en japonais, permet les conversions et les opérations inter-bases dans les bases 2,10 et 16 (pas d'octal donc) avec gestion du complément à 2.



Ce module se charge dans la zone basic (et donc prévient d'un éventuel écrasement), ça sens un peu l'ajout à la dernière minute leur truc!! Du genre : merde on a oublié les bases et ça part en fab la semaine prochaine, déjà que l'on va être juste pour le module de STAT !!

Les fonctions trigo hyperboliques n'ont pas de touches dédiées il faudra donc frapper directement leurs fonctions respectives (HCS, HSN par exemple en Basic, leur équivalent existant aussi en C).

Notez que la précision interne de calcul est de 12 chiffres avec 10 d'affichés/stockés dans les variables, du traditionnel chez Sharp, pourtant quelques modèles offraient une précision à 20 chiffres en double précision. Je le répète le crédo du Sharp ce n'est pas les maths.

Ne cherchez pas non plus aide aux dérivés, intégrales, calculs financiers et autre études et graphiques de fonctions, ça sera à développer selon besoin.





### Alors je programme en quoi ?

Le PCG850V et le plus abouti de la lignée initiée en 1988 par le SHARP PC E200 comprenant entre autre le G815(1993), G820(1996), G830(1996), G850(1996), le G850S(2000) et même le VS en 2009.

Il est avant tout dédié à l'apprentissage de la programmation et pour celui qui veut s'initier, voire s'améliorer, c'est LA machine à posséder.

Avec cinq langages (BASIC, C, CASL, Assembleur, PIC), un moniteur, un éditeur, un mode stat (en japonais), un mode base (en japonais aussi), une gestion de fichiers, un port communiquant RS232 et un port SIO .... Pardonnez du peu.

On peut dire que c'est l'OP le plus complet (son concurrent le Casio Z1GRa voit son C moins puissant, moins rapide et moins standard, pas de PIC...).

```

150 SWITCH LE
151 CASE 1
152 RESTORE 5100
153 CASE 2
154 RESTORE 5200
155 CASE 3
156 RESTORE 5300
157 CASE 4
158 RESTORE 5400
159 CASE 5
160 RESTORE 5500
161 CASE 6
162 RESTORE 5600
163 ENDSWITCH
  
```

## Le Basic

Pour aborder la programmation, le Basic est certainement le plus approprié. Il est ici au standard Sharp reprenant les principales fonctionnalités de la série 13 et suivante, avec cependant quelques différences :

- La suppression des GOTO, GOSUB calculés, Le RENUM ne les gérant pas. Par contre les étiquettes peuvent être utilisées sous deux syntaxes différentes soit "ETIQUETTE" soit \*ETIQUETTE.  
Ex. : 20 GOTO ETIQ .... 100 \*ETIQ :suite ...
- Pas de piezo intégré donc pas de son malgré l'existence de la l'instruction BEEP (il faudra en effet jouer soit du fer à soudé, soit se reprendre sur le RS232 pour lui donner la parole).
- Un MEM qui devient FRE.
- Le PRINT devient compatible avec le graphisme (auparavant le PRINT effaçait l'ensemble de la ligne sur la serie 13).
- Pas d'AREAD, car pas de DEF
- Gestion du port série différente
- Pas de CLOAD, CSAVE (remplacé par BLOAD et BSAVE).

- Pas de MERGE ni de CHAIN.
- Pour l'ARUN on peut aussi se torch....

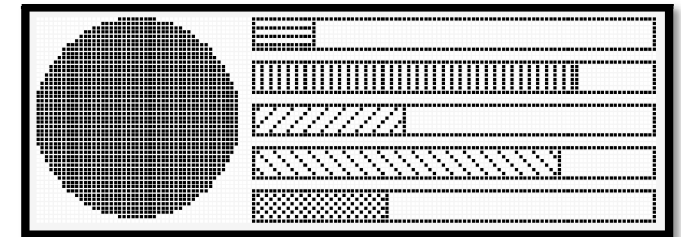
Des structures évoluées apparaissent tel que

- WHILE~WEND, exécutant la boucle pendant la véracité de l'expression. Son pendant.
- REPEAT~UNTIL tournant jusqu'à l'expression.
- Le sélecteur SWITCH~CASE~DEFAULT~ENDSWITCH offrant une alternative au traditionnel ON~GOTO ou ON~GOSUB et traitant en plus les variables alphanumériques.
- Ainsi qu'un IF THEN ELSE ENDIF multi ligne, mais sans possibilité d'utiliser ELSEIF. Offrant ainsi l'accès à la programmation dite structurée.

Les boucles FOR~TO~STEP NEXT, gérant du décimal au niveau du STEP, acceptent le NEXT sans désignation de variables.

Le GOSUB est le seul accès aux sous-programmes mais avec une profondeur de pile de 10 (pour la récursivité on utilisera le C).

Pour les fonctions et procédures utilisateurs, on passe son chemin, ici ni de DEFFN, ni de variables locales. Il est clair qu'ici le basic n'est pas la partie centrale de la machine.



### Une gestion graphique complète avec

- GPRINT (sans filtre binaire like PC 1600 malheureusement) acceptant toujours une chaîne de caractères ou une suite de valeurs numériques séparées par des points virgule.
- POINT retournant 0 ou 1 suivant l'état d'un point de la matrice.
- Le couple PSET et PRESET allumant ou éteignant un point. GCURSOR positionnant le curseur graphique.
- LINE (syntaxe 13xx) pour les lignes, les rectangles vides ou pleins avec paramétrage du type de trait.
- CIRCLE, les cercles pleins ou vides et type du trait.
- Et, nettement plus inédit, PAINT permettant de colorier une surface de 6 motifs différents.

Étonnant toutes ces fonctions sur un si petit écran ! (quoique le X07 à l'époque ait ouvert le bal avec un CIRCLE en 120x32 points).

Des fonctions binaires complètes avec AND, OR, NOT bien sur mais aussi XOR et même INP, nettement moins courant mais sans leur équivalent logique, il faudra donc faire attention pour l'adaptation de programme.

La plage de fonctions mathématiques à orientation scientifique est complète avec les logarithmes en base e et 10 et leurs inverses, la trigonométrie et inverse dans sa version normale et hyperbolique, les permutations et les arrangements, conversions diverses .... Etc., manque à l'appel des fonctions statistiques.

Ses autres domaines de compétences sont le traitement des chaînes de caractères (on reste sur le minimum syndical ici, pas de INSTR, SUB...), la gestion des E/S (RS232 et SIO), Les conversions d'unités (Base, sexagésimal, Système repérage.), le langage machine avec POKE, PEEK, IMP, OUT, CALL sans paramètre autre que l'adresse malheureusement (oh qu'il est bien mon PC1600).

La gestion des Variables est dans le type de la série 125\* et suivantes, variables 7 bits pour les variables fixes A~Z avec un choix imposé numérique ou alpha.

Variables 16 bits pour les textuelles bi-lettres (ab, ac...) avec uniquement deux caractères significatifs, ça fait un peu léger mais sur un pocket y a-t-il besoin de plus ?.

Pour les variables chaînes plus conséquentes c'est avec l'habituelle déclaration de tableau chaîne DIM VA(x)\*y que l'on procède, VA étant le nom de la variable, x le nombre d'éléments et y le nombre de caractères -1 réservés dans la chaîne avec un maxi de 256 caractères (c'est quand même mieux que les 80 habituel). Tableaux, au passage, gérant jusqu'à 2 dimensions.

J'ajoute que les variables BASIC sont permanentes et mémorisées après extinction.

Pour la rapidité du basic il se situe dans le haut du pavé avec la traditionnelle boucle FOR~NEXT à 10000 s'exécutant en 10,5s soit deux fois plus rapide qu'un E500, à relativiser pour les fonctions mathématiques bien sur, et jeu égal avec le 890P pour ce test, mais il écrase son concurrent 16 bits sur 90% des autres tests.

L'éditeur basic est ici pleine page (contrairement à la série 13 mais comme le 1600) il gère les abréviations, vérifie la syntaxe sur validation de la ligne et accepte même les minuscules qui

Basic						
^	&H	ACS	AHC	AHS	AHT	ASN
AND	ASC	ATN	AUTO	BEEP	BLOAD	BLOAD M
BLOAD ?	BSAVE	BSAVE M	CALL	CHR\$	CIRCLE	CLEAR
CLOSE	CLS	CONT	COS	CUB	CUR	DATA
DEG	DEGREE	DELETE	DIGIT	DIM	DMS	DMSS\$
E ND	EOF	ERASE	EXP	FACT	FILES	FIX
FOR~NEXT~STEP		FRE	GCURSOR	GOSUB~RETURN		GOTO
GPRINT	GRAD	HCS	HEX\$	HSN	HTN	IF~THEN~ELSE
IF~THEN~ELSE~ENDIF		INP	INKEY\$	IMP	INPUT	INPUT#
INT	KILL	LCOPY	LEFT\$	LEN	LET	LFILES
LINE	LIST	LLIST	LN	LNINPUT#	LOAD	LOCATE
LOF	LOG	LPRINT	MID\$	MDF	MON	NCR
NEW	NOT	NPR	ON~GOTO	ON~GOSUB	OPEN	OR
OUT	PAINT	PASS	PEEK	PI	PIOGET	PIOPUT
POINT	POKE	POL	PRESET	PRINT	PRINT->LPRINT	PRINT#
PSET	RADIAN	RANDOMIZE	RCP	READ	REC	REM
RENUM	REPEAT~UNTIL	RESERVED	RESTORE	RIGHT\$	RND	RUN
SAVE	SGN	SIN	SQR	SQU	STOP	STR
SWITCH~CASE~DEFAULT~ENDSWITCH			TAN	TEN	TROFF	TRON
USING	VAL	VDEG	WAIT	WHILE~WEND	XOR	

seront converties, les lignes vont jusqu'à 255 caractères.

La saisie et correction est largement facilitée par la présence de cette touche [BACKSPACE] et les modes d'insertion et superposition précédemment évoqués.

Il ne lui manque que le copier/coller (ça existe bien sur le Casio 850P et suivant par le MEMO).

Les possibilités de renommer les lignes et de les numéroter automatiquement sont présentes par les fonctions RENUM et AUTO.

Le TRON et TROFF font parti du jeu pour le débogage, secondé par une notification automatique de l'erreur en pointant sur la ligne en question, il manque par contre la gestion des erreurs (pas de ON ERROR, ERR, RESUME NEXT et consort).

# STRUCTURES

**IF condition THEN**

*instructions*

**[ELSE]**

*instructions*

**ENDIF**

**REPEAT**

*instructions*

**UNTIL condition**

**WHILE condition**

*instructions*

**WEND**

**SWITCH variables**

**CASE valeur**

*instructions*

**[CASE valeur**

*instructions]*

**[DEFAULT**

*instructions]*

**ENDSWITCH**

# L'éditeur

L'éditeur intégré est accessible par la touche **TEXT**. Plus qu'un éditeur c'est une gestion de fichier complète permettant bien sur de générer des fichiers mais aussi de les sauvegarder en



les nommant, de les imprimer, de les envoyer ou recevoir sur le RS232, de les sauvegarder en les nommant de passer des sources entre l'éditeur et le Basic et même la gestion d'un RAMDISK il devient rapidement évident que les 32 Ko de RAM non extensible ne suffiront pas et qu'il faudra utiliser à outrance la connexion RS232.

À noter qu'un cordon PCG850/USB existe et permet de plus d'avoir du son.

Un système de menu et sous menu dont la fonction est appellable par la lettre en majuscule du mot clé permet d'avoir accès à 17 fonctions

**L'éditeur** proprement dit est accessible par le menu principal en appuyant sur la touche E, c'est un éditeur pleine page avec pour seule contrainte, et non des moindres ... d'avoir des numéros de ligne. Après avoir essayé de trouver une explication rationnelle à ce fait et une quelconque utilité il s'avère que c'est d'une inutilité totale (le Casio 890P sans sort très bien sans) mais de plus ça fait consommer de la RAM inutilement et pourtant il y en a peu (j'insiste il est vrai, mais 32Ko c'est 10ko de développement C maxi se répartissant entre la sauvegarde, le source et le compilé).

L'éditeur à outre les fonctions d'édition de l'éditeur basic, avec le mode insertion et superposition, quelques fonctions intégrées :

- A[xx],[yy] pour l'auto numérotation des lignes.
- Lxx pour le listage.
- R[xx],[yy],[zz] pour la renumérotation.
- Dxx,[yy] pour l'effacement.
- Cxx,yy,zz pour la copie d'une ou plusieurs lignes.

- S[0]1,"chaîne" pour la recherche d'une expression.
- E[0]1,"chaîne1","chaîne2" pour le remplacement d'une chaîne par un autre.

Par exemple un simple C100,300,2000 permettra de copier les lignes de 100 à 300 sur la ligne 2000 et suivantes ... pas mal quand même et inexistant sur l'éditeur Basic.

Ou un simple E"Varia","va" permettra de changer la chaîne sur l'ensemble du source.

A noté aussi la touche [TAB] pour l'indentation non automatique des sources assembleur et CASL, pour le C l'écran est trop petit on utilisera des espaces (d'où ma remarque précédente sur les 40 caractères souhaités). La suppression du source est possible dans le menu principal de la touche [TEXT].

L'impression est possible sur CE-126P se connectant sur le port I/O.

Avec le sous menu **SIO** c'est les communications que l'on gère paramétrable à souhait :

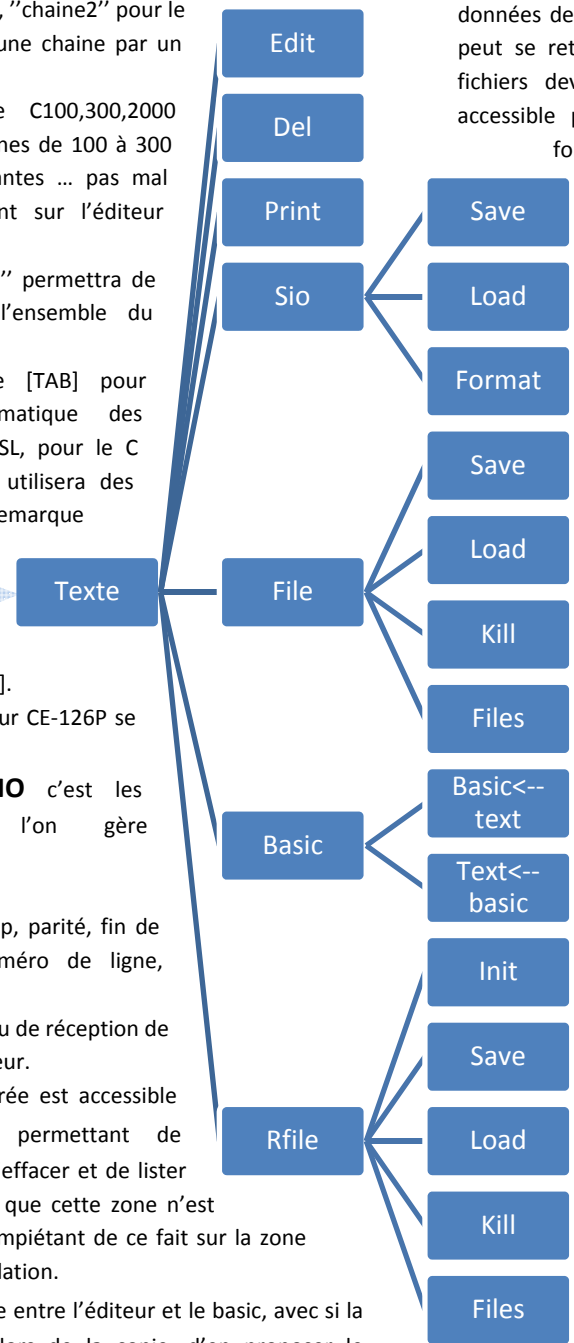
Baud de 300 à 9600

Nombre de bits, Bit de stop, parité, fin de ligne, fin de fichier, numéro de ligne, protocole.

Avec la possibilité d'envoi ou de réception de fichier depuis ou vers l'éditeur.

La gestion de fichier intégrée est accessible depuis le menu **File** permettant de sauvegarder, de charger, d'effacer et de lister les fichiers existant. Notez que cette zone n'est pas fixe mais dynamique empiétant de ce fait sur la zone programme, éditeur, compilation.

**Basic** le pont indispensable entre l'éditeur et le basic, avec si la mémoire est insuffisante lors de la copie, d'en proposer le transfert.



La dernière option **Rfile** permet la création d'un fichier de données de taille fixée, donc attention lors du paramétrage on peut se retrouver avec un manque de place. Ce fichier (ces fichiers devrais-je dire car il peut en avoir plusieurs) est accessible par programmation, en basic et en C, avec les fonctions OPEN, PRINT# et INPUT# pour le basic.

On peut aussi à travers ce menu les sauvegarder, les charger, les effacer et les lister.

On entend par sauvegarde et chargement une relation directe avec l'éditeur, mais il est évident que ces RFILE ne sont pas dédié à la sauvegarde des sources mais bien à la sauvegarde de données, d'ailleurs l'extension implicite .DAT ne trompe pas. Le passage par l'éditeur est justifié pour une éventuelle impression ou communication RS232.

## GESTION DES RFILES

Après un INIT RFILES, FRA.DAT

### En Basic

```
5 A=5
10 OPEN "E:FRA.DAT" FOR OUTPUT AS #2
20 PRINT #2,A
30 A=2
40 CLOSE
50 OPEN "E:FRA.DAT" FOR INPUT AS #2
60 INPUT #2,A
70 PRINT A
80 END
```

### En C

```
10 void print(char nom,int x,int y){
20     printf("%c %d %d\n",nom,x,y);
30 }
40 main(){
50     char nom=125;
60     int x=1,y=2;
70     FILE * sortie, * entree;
80     sortie=fopen("FRA.DAT","w");
90     fprintf(sortie,"%c%d %d\n",nom,x,y);
100    nom=145,x=17,y=18;
110    print(nom,x,y);
115    fclose(sortie);
120    entree=fopen("FRA.DAT","r");
130    fscanf(entree,"%c%d%d",&nom,&x,&y);
140    print(nom,x,y);
200 }
```

# Le C

Le langage C sur un ordi de poche c'est peu courant, en fait en dehors du Japon c'est même inédit. Le C embarqué dans le Sharp PCG850V est le plus puissant embarqué sur ordi poche, il est aussi le plus rapide et le plus standard (j'insiste et je vise en particulier le concurrent de chez Casio).

Si le Basic avait une amorce de structure ici avec le C on est en plein dedans, variables globales, locales, constantes, pointeurs, fonctions, procédures. Tout le C standard y est, même les structures absentes du Casio (j'ai re-dis Casio ???).

En plus des fonctions C standard Sharp à ajouté des fonctions mathématiques (en bleu dans le tableau) et des fonctions graphique (en rouge) et même, planqué en vert, le son si la modification précédemment citée est posée.

Pour qui veut se mettre au C c'est vraiment l'outil idéal. Petit, boot instantané, compilation instantané. Le G850V vous permettra de vous entraîner à la pratique du C tant dans le métro que durant la pose de midi, ou même en vacances (surtout en vacances).

Le manuel, en japonais au demeurant, est très bien fait et didactique, même sans aucune connaissance de la langue du pays du soleil levant on s'en tire bien, quelques outils de traduction peuvent lever l'ambiguïté sur certains points.

Les premiers pas sur ce langage pour le non habitué, tel que moi, c'est d'essayé de retrouver ses nouveaux points de repère.

Pas si évident que cela pour celui qui pense Basic, et envisage la traduction mot à mot. Le but est d'apprendre à penser C (je n'aurais pas voulu la rater celle là)!

Certaines structures comme IF~ELSE, FOR, DO~WHILE ... oups je parle en majuscule là, C est sensible à la casse !! (la sale bête), je disais donc, certaines structures comme if~else, for, do~while, switch ont leur analogie Basic, les fonctions, procédures et portée des variables ont leur équivalent Pascal reste les pointeurs qui ne pose que peu de problème de compréhension à celui qui a déjà pratiqué le langage machine ou l'assembleur. Reste donc à assimilé la syntaxe propre au C et surtout oublier les autres langages pour apprécier à sa juste valeur ce dialecte. Et on se prend vite au jeu, à tel point que le C pourrais être le langage du pocket par excellence.

**Aller** on se lance, pour la programmation en C on doit d'abord taper son source dans l'éditeur de texte vue précédemment. Ensuite la procédure est de le compiler.

Pour cela il y a la touche TEXT shiftée qui donne accès à quatre

```
<<< C-LANGUAGE >>>
copyright(c) 1995 by
SHARP CORPORATION /
FIRMWARE SYSTEMS, INC.
```

```
*** C ***
Compile Trace Go Stdout
```

options (voir graphique complet page suivante):

- La compilation du source.
- Le mode trace une exécution pas à pas.
- Le lancement du programme compilé.
- La désignation du périphérique standard de sortie.

La **compilation** du source ne pose pas de problème majeur.

Un simple appui sur 'C' le compile. Les directives #define, #include, #if~\$ elid~# else~\$ endif et #ifdef \$ ifndef sont impeccablement gérées.

Si un problème intervient lors des passes du compilateur le numéro de ligne contenant l'erreur sera indiquée ainsi que une explication sommaire mais suffisamment explicite de celle-ci. Exemple : '(870) undefine : x' pour problème à la ligne 870, variable x non déclarée.

Un retour rapide sur l'éditeur, un Lxxx (aller à la ligne

xxx) et une correction réglera rapidement le problème (enfin dans le meilleur des mondes). J'attire l'attention sur un oubli du point virgule (courant pour les habitués du Basic) sur la fin de ligne donnera un message d'erreur sur la ligne suivante.

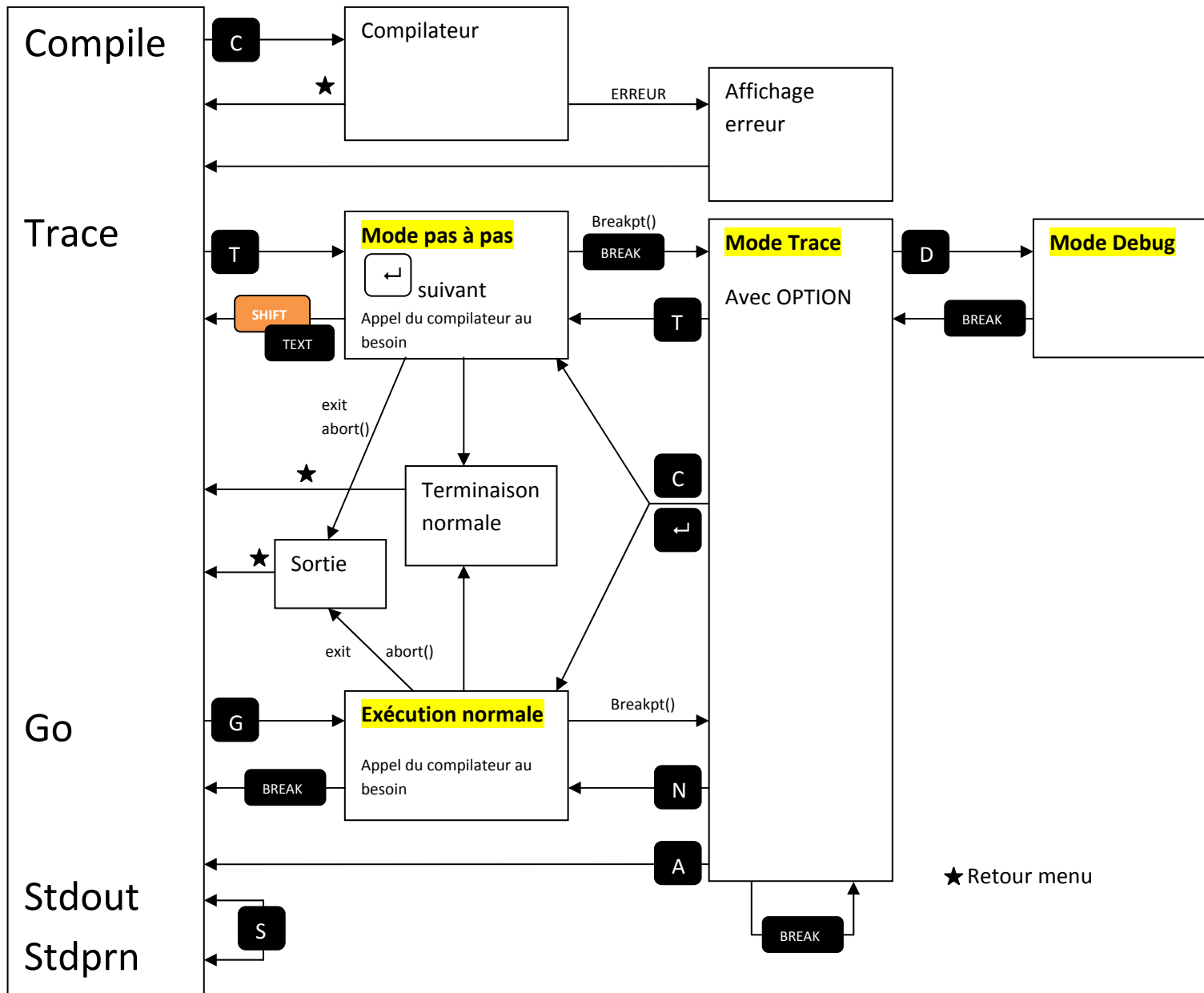
Le mode **Trace** permet l'accès à plusieurs options par la lettre majuscule.

- **[ENTER]** Ligne suivante
- **C** pour continu en mode breakpt et retour au pas à pas en trace.
- **A** pour le retour au menu principal.
- **T** pour le retour au pas à pas.
- **N** pour l'exécution de la suite du programme en mode normal.
- **D** pour visualiser le contenu d'une variable, ça c'est bien pratique !

Le **Go** est simplement l'exécution du compilé et si besoin la compilation du source, si un breakpt() est rencontré on atteint un menu permettant au besoin de continuer.

La dernière option : Stdout, faute du périphérique adéquat (CE 126P), je ne puis tester mais ça paramètre le périphérique de sortie par défaut (si une âme charitable m'en passe une je

C						
#define	#include	#if~\$ elid~# else~\$ endif	#ifdef \$ ifndef	++	--	
&	&&		!	^	~	
<<	>>	?	abs	acos	acosh	
angle	asin	asinh	atanh	auto	beep	
break	breakpt	call	char	circle	clearerr	
clrscr	const	cos	continue	do~while	double	
enum	exp	exit	fclose	feof	fflush	
fclose	fgetc	fgets	fopen	fprintf	free	
fputc	fputs	fscanf	float	for	gcursor	getc
getchr	getchar	gets	goto	gotoxy	gprint	if
if~else	inport	int	isalnum	isalpha	iscntrl	isdigit
isgraph	islower	islower	isprint	ispunct	isspace	isupper
isxdigit	kbhit	line	log	log10	long	main
malloc	miniget	miniput	outport	paint	peek	pioget
pioput	pioset	poke	point	pow	preset	printf
pset	put	putchar	puts	register	return	scanf
sscanf	signed	sin	sinh	sizeof	sprintf	sqrt
static	strcat	strchr	strcmp	strcpy	strlen	struct
switch~case~default~break	tan	tanh	tolower	toupper	typedef	
union	unsigned	void	volatile	while		



**Le source** est donc frappé sous l'éditeur, que l'on a vu précédemment. Ici, contrairement au mode BASIC, ne cherchez pas une vérification syntaxique à la validation de ligne. Les éventuelles erreurs seront signalées qu'à la compilation.

Un programme C est une suite de déclaration de fonctions et de procédures avec une désignation précise des variables d'entrées et du retour dans le cas d'une fonction. La procédure est une fonction particulière qui ne retourne rien (précédé de *void*).

Une procédure essentielle à l'exécution du programme est la procédure *main()* servant au lancement, Sans elle pas d'exécution possible, votre programme ne sera qu'une bibliothèque de fonction 'Mergeable' avant compilation avec un autre avec la directive `#include`.

Les variables apportent aussi plein de nouvelles notions par rapport au Basic. On retrouve bien sur les variables numériques, les variables alphanumériques (avec une notion très différente du Basic) et leurs déclinaisons tabulaires. Mais en C ça devient beaucoup moins souple. Elles doivent être déclarées et ont une portée soit limitée à une fonction, soit à l'ensemble du programme.

Pour les variables numériques c'est plus de 10 types différents que l'on peut gérer en comptant leurs déclinaisons signées, non-signées et les types présent pour la compatibilité.

TYPE	étendue
char	-128 ~ +127
unsigned char	0 ~ 255
short	-32768 ~ +32767
Unsigned short	0 ~ 65035
Int	-32768 ~ +32767
Unsigned int	0 ~ 65035
Long	-2147483648 ~ +2147483647
Unsigned long	0 ~ 4294967295
Float	±1E-99 ~ ±9.999E+99
double	±1 E-99 ~ ±9.999999999E+99
Long double	±1 E-99 ~ ±9.999999999E+99

Bien sur les fonctions de conversion inter-type sont implémentées.

On peut aussi gérer des constantes, des variables non modifiables en somme (étrange ...), et des statiques qui reste à leur valeur après la sortie de fonction, avec **const** et **static**.

pourrais compléter ce point).

Le C étant compilé (just in time) avant son exécution il faudra, indépendamment des variables déclarées, près de deux fois plus de mémoire que la taille du programme source pour son exécution. Autant dire que lorsque l'on bosse sur la création

d'un programme en C avec sa sauvegarde, c'est vers 10Ko que l'on commence à avoir des problèmes. On pourra toujours ruser par un `#INCLUDE`. Sachant en plus que de 10 à 20% du source du programme passe en espaces et numéros de ligne, on se sent très à l'étroit.

La trame d'un programme simple sera :

```
Main(){
  Déclaration variables locales
  ...code...
}
```

Les fonctions et procédures peuvent être décrites avant ou après le **main()**, mais dans le cas d'un codage postérieur, une déclaration au sein du main() sera obligatoire (le terme est prototype).

Exemple déclaration de fonctions et procédures (void en tête = procédure)

```
50 main(){
60 unsigned char code,ascii;
70 /*prototype*/
80 double val(char*);
90 unsigned int key(unsigned char*,unsigned char*);
100 char* stredit(char str[],unsigned char asc);
110 void calcul();
120 char fonction(unsigned char);
130 void pile(char);
140 void interf();
150 void affpile();
```

Fonction et procédures peuvent accepter des paramètres. Ces paramètres peuvent être des variables (*char fonction(unsigned char)*) ou des tableaux de variables (*stredit(char str[],...)* déjà connu en BASIC et même des pointeurs permettant de faire du référentiel (*key(unsigned char\*,unsigned char\*)*).

L'objet est présent sous la forme des structures et des unions (pas sur le Casio Z1 ça).

### Comparaison

a==b	égalité
a !=b	Différent
a<b	Strictement inférieur
a>b	Strictement supérieur
a<=b	Inférieur ou égal
a>=b	Supérieur ou égal

### Affectation

a=b	Affectation de la valeur b à a
a+=b	Ajout à a de b (a=a+b)

a-=b	Soustrait de a, b (a=a-b)
a*=b	Multiplie a par b (a=a*b)
a/=b	Divise a par b (a=a/b)
a%=b	Modulo de a par b (a=a%b)

### Logique

a&& b	Et logique
a     b	Ou logique
!a	Not logique

### Binaire

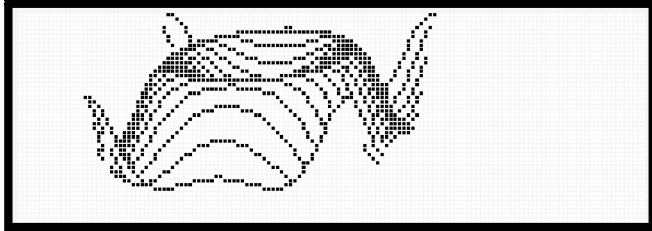
A&b	Et binaire
a   b	Ou binaire
a ^ b	Ou exclusif binaire
~a	Not binaire
a << b	Décalage binaire à gauche de b bits
a >> b	Décalage binaire à droite de b bits



Une adaptation d'un programme Basic pour Sharp PC 1600/1350 et similaire publiée il y a plus de 25ans dans un mensuel dédié aux ordinateurs Sharp, son auteur un certain M.CHOUKROUN que je salue.

La fonction est modifiable ligne 20

Longueur et hauteur conseillées 80x40



## COURBES

```

10 float fonction(float x,float y){
20   return sin(x*x+y*y);
30 }
40 void liner(int x,int y){
50   line(peek(31079),peek(31081),x,y,0,65535,0);
60 }
70 main(){
80   float *m,*n,h,o,q,s,t,u,x,y,z,mc;
90   float a,b,c,d,e,f,v,w,l,r,p,nc;
100  int i,j,k;
110  angle(1);
120  printf("longueur:");
130  scanf("%f",&l);
140  printf("hauteur:");
150  scanf("%f",&v);
160  printf("decalage:");
170  scanf("%f",&w);
180  m=(float*)malloc(l*sizeof(float));
190  n=(float*)malloc(l*sizeof(float));
200  q=v-40-w;
210  for(i=0;i<=l;i++){
220    n[i]=v;
230  printf("nbre de coupes:");
240  scanf("%f",&nc);
250  printf("x min:");
260  scanf("%f",&a);
270  printf("x max:");
280  scanf("%f",&b);
290  printf("y min:");
300  scanf("%f",&c);
310  printf("y max:");

```

```

320  scanf("%f",&d);
330  printf("z min:");
340  scanf("%f",&e);
350  printf("z max:");
360  scanf("%f",&f);
370  r=1;s=1;
380  p=nc-1;t=(l-r*p)/(d-c);u=(v-s*p)/(f-e);
390  clrscr();
400  for(i=0;i<=p;i++){
410    x=b-(b-a)*i/p;o=0;
420    for(j=0;j<=(l-r*p);j++){
430      y=c+j/t;k=i*r+j;
440      z=fonction(x,y);
450      if(z>=e&&z<=f){
460        mc=u*(z-e);h=(int)(v-i*s-mc+.5);
470        if(h<m[k]&&h>n[k]){
480          o=0;
490        } else {
500          if(h>m[k])
510            m[k]=h;
520          if(h<n[k])
530            n[k]=h;
540          if(o==1) {
550            liner(15+k,h-q);
560          } else {
570            poke(31079,15+k);poke(31081,h-q);
580            o=1;
590          }
600        }
610      }
620    }
630  }
640  gotoxy(20,5);
650  printf("fin");
660  i=getch();
670 }

```

## COPIE ECRAN

```

1000 void hardcpy(){
1010  int a,b;
1020  FILE * sortie;

```

```

1030  sortie=fopen("stdaux","w+");
1040  for(b=0;b<48;b++)
1050    for(a=0;a<144;a++)
1060      fprintf(sortie,"%d",point(a,b));
1070  fclose(sortie);
1080 }

```

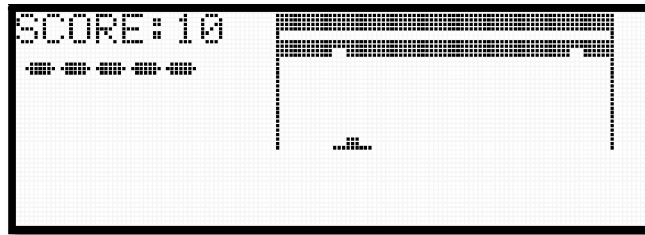
```

8000 OPEN "COM1:"
8010 FOR B=0 TO 47
8020 FOR A=0 TO 143
8030 PRINT #1, STR$ POINT (A,B);
8040 NEXT
8050 NEXT
8060 CLOSE#1
8070 RETURN

```

## VERSION BASIC

```
10 CLS
20 S=1
30 PRINT "SCORE:0"
40 DIM VI$(5)*80
50 RA$="0207070707070200
60 FOR A=1 TO 5
70 VI$(A)=VI$(A-1)+RA$
80 NEXT
85 VI$(0)="0000000000000000
90 GCURSOR (2,18)
100 GPRINT VI$(5)
110 LINE (59,0)-(59,30)
120 LINE (135,0)-(135,30)
130 LINE (60,0)-(134,3),BF
140 LINE (60,6)-(134,9),BF
150 LV=5:SC=0
160 BX=96:OBX=BX
170 Y=10:OX=X:OY=Y:RX=-3:RY=2:RX=(2* RND (2)-3)*RX:X=99-RX*( RND (6)+4)
180 LINE (BX,30)-(BX+8,30)
190 X=X+RX:Y=Y+RY
200 IF X<60 OR X>132 BEEP S,10,10:RX=-RX:X=X+2*RX
210 IF Y<30 THEN 250
220 IF (BX<=OX) AND OX<BX+9 BEEP S,20,10:RY=-RY:Y=OY+RY:GOTO 250
230 IF (OX+RX=BX) OR X-RX=BX+9 BEEP S,20,10:RY=-RY:RX= RX:Y=OY+RY:X=OX+RX:
GOTO 250
240 GOTO 410
250 IF Y=-2 BEEP S,20,10:RY=-RY:Y=OY+RY
260 IF POINT (X,Y)=0 THEN 310
270 BEEP S,10,10
280 SC=SC+5:LOCATE 6,0:PRINT STR$ SC
290 LINE (X-RX,Y)-(X-RX+2,Y+1),R,B
300 Y=Y-2*RY:RY=-RY
310 GOSUB 360
320 LINE (OX,OY)-(OX+2,OY+1),R,B
330 LINE (X,Y)-(X+2,Y+1),B
340 OX=X:OY=Y
350 GOTO 190
360 A$= INKEY$ :A= ASC A$
370 BX=BX+6*(A=13)*(BX<123)-6*(A=32)*(BX>63)
380 LINE (OBX,30)-(OBX+8,30),R:LINE (BX,30)-(BX+8,30)
390 OBX=BX
400 RETURN
410 LINE (OX,OY)-(OX+2,OY+1),R,B
420 LINE (X,Y)-(X+2,Y+1),B
430 LV=LV-1
440 GCURSOR (2,18):GPRINT VI$(LV);"0000000000000000"
450 FOR P=1 TO 255 STEP 3:BEEP S,P,2:NEXT
460 LINE (X,Y)-(X+2,Y+1),R,B
470 LINE (BX,30)-(BX+8,30),R
480 IF LV>0 FOR P=1 TO 1000:NEXT :GOTO 160
490 END
```



## Comparaison syntaxique BASIC/C sur le même programme avec une programmation BASIC type non-structurée.

En C le INKEY\$ a été réalisé en assembleur car la fonction n'existe pas, l'appel étant réalisé par CALL avec passage de paramètres, fonction sans équivalence BASIC.

La fonction RND est aussi inconnue en C et seule la création d'une routine de génération de nombre pseudo-aléatoire pourra résoudre ce manque.

A noter aussi la vitesse ralentie par la ligne 260 en C car ça va trop vite. Avec nouvelles commandes BASIC WHILE~WEND, IF~ENDIF on pourrait, comme le programme en C, se passer des GOTO déstructurant et avoir un code similaire.

## VERSION C

```
10 #define M 65535
20 #define VIT 200
30 #define INKEY call((int)ink,0)
40 void vie(int nbr)
50 {
60 int b;
70 gcursor(2,18);
80 for(b=1;b<=nbr;b++)
90 gprint("0207070707070200");
110 gprint("0000000000000000");
120 }
130 main()
140 {
150 char *ink="\xCD\x53\xBE\x6F\x26\x00\xC9";
160 int vi=5,sc=0,x=96,y=10,ox=x,oy=y,rx=3,ry=2,bx=x,obx=x,b;
170 printf("Score:0");
180 line(59,0,59,30,0,M,0);
190 line(135,0,135,30,0,M,0);
200 line(60,0,134,3,0,M,2);
210 line(60,6,134,9,0,M,2);
220 vie(vi);
230 do
240 {
250 line(bx,30,bx+8,30,0,M,0);
260 for(b=1;b<VIT;++b){}
270 x+=rx;y+=ry;
280 if(x<60|x>132)
290 {
300 beep(10,10,1);
310 rx=-rx;x+=2*rx;
320 }
330 if(y>=30)
340 {
350 if(bx<=ox&&ox<bx+9)
360 {
370 beep(10,20,1);
380 ry=-ry;y=oy+ry;
390 }else if (ox+rx==bx| |x-rx==bx+9)
400 {
410 beep(10,20,1);
420 ry=-ry;rx=-rx;y=oy+ry;x=ox+rx;
430 }else {
440 line(ox,oy,ox+2,oy+1,1,M,1);
450 line(x,y,x+2,y+1,0,M,1);
460 vie(-vi);
470 for(b=1;b<256;b+=3)
480 {
490 beep(2,b,1);
500 }
510 line(x,y,x+2,y+1,1,M,1);
520 line(bx,30,bx+8,30,1,M,0);
530 x=96;y=10;ox=x;oy=y;rx=3;ry=2;bx=x;obx=x;
540 }
550 }else {
560 if(y== -2)
570 {
580 beep(10,20,1);
590 ry=-ry;y=oy+ry;
600 }
610 if (point(x,y))
620 {
630 beep(10,10,1);
640 sc+=5;
650 gotoxy(6,0);printf("%d",sc);
660 line(x,y,x+2,y+1,1,M,1);
670 y=-ry;ry=-ry;
680 }
690 b=INKEY;
700 bx=bx+6*(b==34)*(bx<123)-6*(b==33)*(bx>63);
710 line(obx,30,obx+8,30,1,M,0);
720 line(bx,30,bx+8,30,0,M,0);
730 obx=bx;
740 line(x,y,x+2,y+1,0,M,1);
750 line(ox,oy,ox+2,oy+1,1,M,1);
760 ox=x;oy=y;
770 }
780 }
790 while(vi>0);
800 }
```

Abort  
int abs(int)  
double acos(double)  
double acosh(double)  
void angle(unsigned)  
double asin(double)  
double asinh(double)  
double atan(double)  
double atanh(double)  
auto  
beep(unsigned longueur, long frequence, unsigned nombre)  
break  
void breakpt(void)  
unsigned call(unsigned adr, void\* arg\_HL)  
void\* calloc(unsigned n, unsigned taille)  
char  
int circle(int x, int y, int r, double angle depart, double angle fin)  
void clearerr(FILE\* flux)  
void clrscr(void)  
const  
double cos(double)  
double cosh(double)  
continue  
do~while  
double  
enum  
exp  
exit  
extern  
fclose  
feof  
fflush  
fclose  
fgetc  
fgets  
flop  
fopen  
fprintf  
free  
fputc  
fputs  
fscanf  
float  
for  
gcursor  
getc  
getchr  
getchar  
gets  
goto  
gotoxy  
gprint  
if  
if~else  
inport  
int  
isalnum  
isalpha  
iscntrl  
isdigit  
isgraph  
islower  
isprint  
ispunct  
issapace  
isupper  
isxdigit  
kbhit  
line  
log  
log10  
long  
main  
malloc  
miniget  
miniput  
outport  
paint  
peek  
pioget  
pioput  
pioset  
poke  
point  
pow  
preset  
printf  
pset  
put  
putchar  
puts  
register  
return  
scanf  
sscanf  
signed  
sin  
sinh  
sizeof  
sprintf  
sqrt  
static  
strcat  
strchr  
strcmp  
strcpy  
strlen  
struct  
switch~case~default~break  
tan  
tanh  
tolower  
toupper  
typedef  
union  
unsigned  
void  
volatile  
while

# Le CASL

L'apprentissage de l'assembleur n'étant pas franchement facile, l'idée des écoles japonaises, en collaboration avec le ministère de l'industrie japonais, était de faire un cahier

des charges d'un processeur fonctionnant dans un environnement protégé, avec un jeu d'instructions établi une fois pour toute. De cette façon aucun plantage de la machine sur une éventuelle erreur n'est possible et la mise en place de travaux pratiques types par les écoles est plus facile.

Le Comet étant implémenté sur plein d'ordinateur de poche différent, dont le G850V.

<b>Directives</b>	START	Début du programme
	END	Fin du programme
	EXIT	Sortie du programme
	IN	Entrée clavier
	OUT	Sortie écran
	WRITE	Affichage des registres et attente d'une touche
	DC	Définition d'une constante
	DS	Définition d'un stockage
<b>Registre</b>	LD GR,adr[,XR]	Chargement d'une valeur pointée
	ST GR,adr[,XR]	Sauvegarde d'une valeur dans une adresse
	LEA GR,val[,XR]	Chargement d'une valeur
	ADD GR,adr[,XR]	Addition d'une valeur pointée
	SUB GR,adr[,XR]	Soustraction d'une valeur pointée
<b>Logique</b>	AND GR,adr[,XR]	Et binaire
	OR GR,adr[,XR]	Ou binaire
	EOR GR,adr[,XR]	Ou exclusif binaire
<b>Comparaison</b>	CPA GR,adr[,XR]	Comparaison arithmétique
	CPL GR,adr[,XR]	Comparaison logique
<b>Décalage</b>	SLA GR,adr[,XR]	Décalage à gauche arithmétique
	SRA GR,adr[,XR]	Décalage à droite arithmétique
	SLL GR,adr[,XR]	Décalage à gauche logique
	SRL GR,adr[,XR]	Décalage à droite logique
<b>Saut</b>	JPZ adr[,XR]	Saut sur positif ou nul
	JMI adr[,XR]	Saut sur négatif
	JNZ adr[,XR]	Saut sur non nul
	JZE adr[,XR]	Saut sur 0
	JMP adr[,XR]	Saut incondtionnel
<b>Pile</b>	PUSH adr[,XR]	Mise sur la pile
	POP GR	Prise d'une valeur sur la pile
<b>Routine</b>	CALL adr[,XR]	Appel d'une routine
	RET	Sortie d'une routine

## Registres

généraux : GR0, GR1, GR2, GR3

Pointeur programme : PC

drapeau : FR

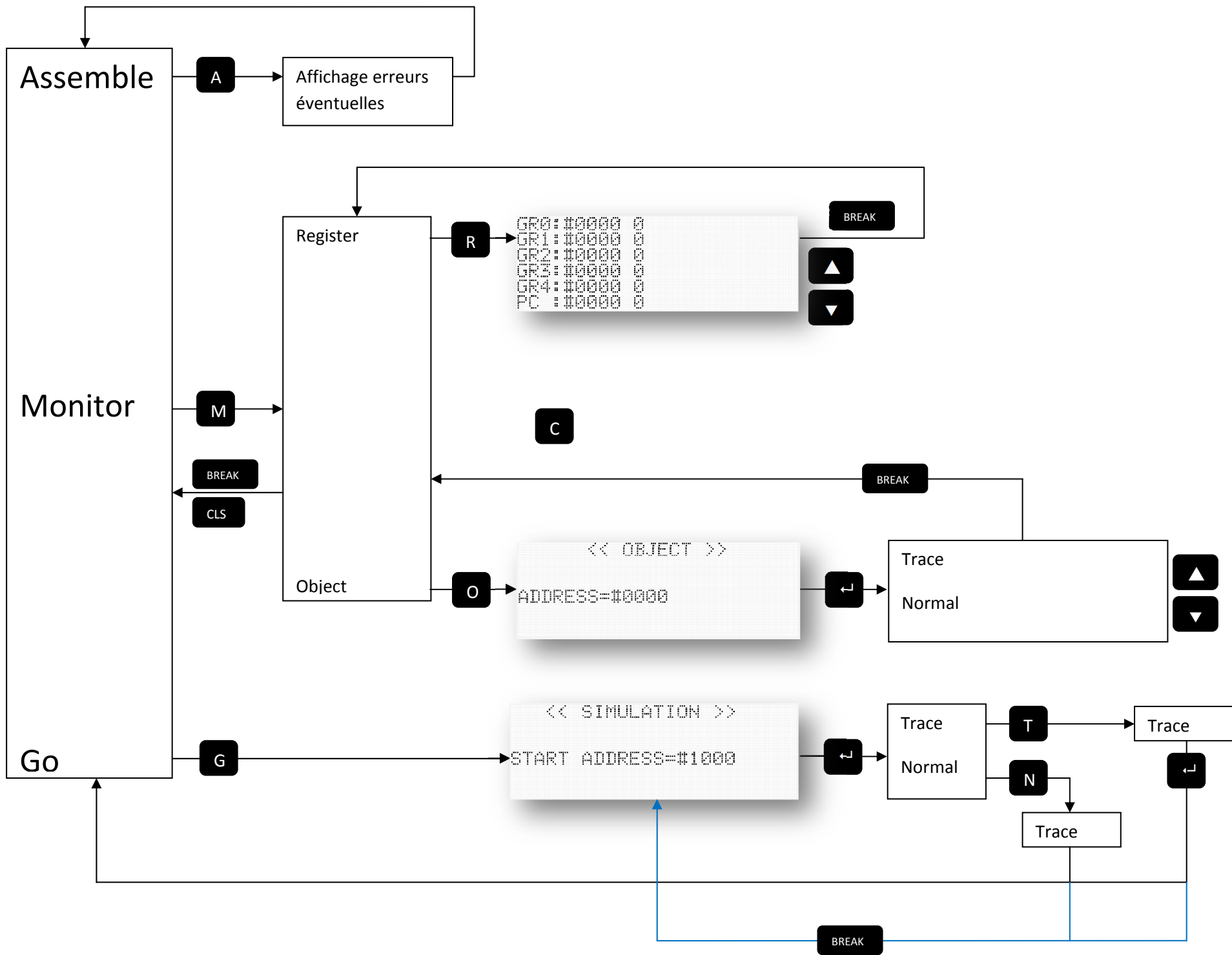
**Utilisé pour contrôler l'exécution de la simulation.**

Pause : BP et Break : BC

## CODAGE

10TEST	START	
20	IN	CAR,NBR
30	LEA	GR1,0
40FOR	LD	GR0,CAR,GR1
50	ST	GR1,COD
60	ADD	GR0,COD
70	ST	GR0,CAR,GR1
80	ADD	GR1,INC
90	CPA	GR1,NBR
100	JMI	FOR
110	OUT	CAR,NBR
120	EXIT	
130INC	DC	1
140COD	DS	1
150CAR	DS	24
160NBR	DS	1
170	END	

SHIFT TEXT , C, A, G, ← , N  
ABCDEFGH ←



# L'assembleur

Passé le stade du

Comet autant finir les

maines dans le cambouis. Là plus de garde

fou, la moindre erreur sera sanctionné par **dans** le mieux un léger blocage, dans le pire un reset total...

Le cœur du G850V est un Z80 donc pas de surprises, c'est de l'archi connu, une connaissance de la structure mémoire de la machine est cependant indispensable pour ne pas taper dans la RAM système, RAM source et basic ou la zone de variable.

Pour cela il faut définir une zone protégé par l'instruction USER saisie sous le MONiteur (voir plus bas) ça permet de réserver une zone non écrasable dédié au LM, cette zone commence à &H100, par exemple un USER200 réservera un zone de &H100 à &H200

**Le moniteur** Accessible dans le mode  
RUN du Basic il se lance avec  
l'instruction MON.....

# TESTS DE RAPIDITE

		Année	Boucle	Goto	Constante	Variable	Addition	Multiplication	Logique	Modulo	Puissance	Trigo	Log	Chaine	Précision	Précision 2	Graphisme	Récurtivité
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
<b>CANON</b>	<b>X07</b>	<b>1983</b>	31,5	11,3	7,2	5,4	7,7	11,9	8,8	10,3	27,4	128	213	9,4	14,14	4,20E-07		MEM
<b>CASIO</b>	<b>FX 702P</b>	<b>1982</b>	176	114	52	50	66	72	NA	NA	328	277	203	84	12,12	7,56E-05	NA	10
	<b>FX 790P</b>	<b>1986</b>	105	49,3	30	31,3	39,5	42,7	NA	NA	170	135,3	108	45,5	12,12	7,56E-05	NA	8
	<b>FX 880P</b>	<b>1990</b>	28	86	28	30	76	76,8	58	61	99,5	64,5	56,5	26	12,12	7,56E-05	NA	95
	<b>Z1-GR</b>		9,9	6,3	3,3	3,1	4,5	5,3	5,3	6,7	47,4	31,7	25,8	5,5	13,13	4,45E-06	152	MEM
	<b>Z1-GR©</b>		11,7		1,8							31,4			13,13	4,45E-06	NA	MEM
	<b>48GX</b>	<b>1996</b>	27	NA	7,4	8,6	11,3	11,8	14,8	12	24,4	24,3	13,5	12,7	12,12	1,74E-05	213	MEM
	<b>35S</b>	<b>2008</b>	375	63,5	39	44,5	69,5	72	153	67,5	99	104	92	NA	12,12		NA	20
<b>SHARP</b>	<b>PC 1500</b>	<b>1981</b>	142	28	20,5	20,3	25	31	26,5	65,7	216	169	149,5	35,5	10,12	5,61E-04	NA	
	<b>PC1251</b>	<b>1982</b>	423														NA	
	<b>PC 1262</b>	<b>1986</b>	75	50	21,5	20,5	30	35,5	47	78,5	260	187	168	51,6	10,12	5,61E-04	NA	10
	<b>PC 1360</b>	<b>1986</b>	82	55	24,6	25,5	36,3	41,5	46	86	265	189	169	59	10,12	5,61E-04		10
	<b>PC 1600</b>	<b>1987</b>	32	12,2	5,3	5,3	7,5	10,4	12,3	37	202	162	141	20,8	10,12	5,61E-04	79	30
	<b>PC G850V</b>	<b>2002</b>	10,5	5,3	2,7	2,2	3,1	4,2	5,5	10	40,1	30,7	32,3	46,7	10,12	5,61E-04	32	10
	<b>PC G850V©</b>	<b>2002</b>	4,1		0,72							29,3			10,12	5,61E-04	9,5	MEM
	<b>PC 1450</b>	<b>1985</b>	64	47	19	18,5	28,5	33,5	45	76,5	259	185,5	164	48,5	10,12	5,61E-04	NA	10
	<b>PC 1246</b>	<b>1984</b>	80	18,5	13	13,3	17,5	19,7	28,7	37	100	93	70	28	10,12	5,61E-04	NA	10
<b>TANDY</b>	<b>MODEL 100</b>	<b>1983</b>	27,9	9,2	6	4,5	6,5	10,1	7,3	8,4	23,1	104	191	7,4	14,14	4,20E-07		
<b>TI</b>	<b>74</b>	<b>1985</b>	48	13,6	8,7	9,3	12,6	14,1	14,4	27	16,5	155	183	18,8	14,14	1,78E-06	NA	MEM
	<b>NSPIRE</b>	<b>2007</b>	2,7	NA	0,49	0,56	0,65	0,68	0,67	0,68	0,77	1,65	1,24		14,14		NA	MEM





**Quelques routines en ROM**

Adresse	Nom	Explication
BCBE	STAT	Appel du mode STAT
BCF1	CLRBAS	BASIC DELETE OK ?
BCF7	CLRTXT	TEXT DELETE OK ?
BCFD (88C1)	GETCHR	Lecture d'un caractère au clavier vers registre A
BD00 (86FA)	LDPSTR	Lecture d'une suite de pixel de VRAM vers HL E position en X D position en Y
BD03	REGOUT	Affichage de tout les registres Z80 et attente d'une touche
BD09	AOUT	Affichage du registre interne A et modification possible
BD0F	HLOUT	Affichage du double registre interne HL et modification possible
BD2D	OFF	POWER OFF, Extinction du Sharp
BE53 (89BE)	INKEY	Lecture du code de touche clavier dans A Si une touche est appuyée alors A=code touche et Carry=true Si pas de touche alors A=0 et Carry=false Si plusieurs touches A=52
BE62 (8440)	PUTCHR	Affichage d'un caractère sur l'écran A code du caractère E position en X D position en Y
BE65	INSLN	Insertion d'une ligne vide à la position (E,D) (24 espace). E position en X D position en Y