

Programmation BASIC SANYO PHC-25



Version du 26/06/2025

Dédicace à tous ceux qui préserve le patrimoine informatique.

Un grand merci à Olipix qui a lancé les GAME JAM.

Table des matières

Table des matières.....	5
Introduction	11
Les variables	13
Le BASIC	15
ABS.....	19
AND.....	20
ASC	21
CHR\$.....	22
CLEAR	23
CLOAD.....	24
CLOAD?	25
CLS.....	26
COLOR	27
CONSOLE.....	36
CONT.....	37
COS.....	38
CSAVE	39
CSRLIN	40
CTOFF.....	41
CTON.....	42
DATA	43
DEF FN.....	44
DIM.....	46
ELSE	47

END	48
EXEC.....	49
EXP.....	50
FRE.....	51
FOR	52
GOSUB	53
GOTO.....	54
IF	55
INKEY\$.....	56
INP	57
INPUT	58
INPUT#	59
INT.....	60
KEY	61
LCOPY	62
LEFT\$.....	63
LEN.....	64
LET	65
LINE.....	66
LIST.....	68
LLIST.....	69
LOCATE.....	70
LOG	71
LPOS	72
LPRINT.....	73
MID\$.....	74

NEW	75
NEXT	76
NOT	77
ON GOSUB	78
ON GOTO	79
OR	80
OUT	81
PAINT	82
PEEK	83
PLAY	84
POINT	86
POKE	87
POS	88
PRESET	89
PRINT	90
PRINT#	91
PSET	92
READ	93
REM	94
RESTORE	95
RETURN	96
RIGHT\$	97
RND	98
RUN	99
SCREEN	100
SCREEN 1	101

SCREEN 2	102
SCREEN 3	103
SCREEN 4	104
SCRIN	105
SGN	106
SIN	107
SLOAD	108
SOUND	109
SPC	110
SQR	111
SSAVE	112
STEP	113
STICK	114
STOP	115
STRIG	116
STR\$	117
TAB	118
TAN	119
TIME	120
THEN	121
USR	122
VAL	123
Mémoire du PHC-25	127
L'écran du SANYO PCH-25	128
Jeu de caractères	141
Trucs et Astuces	143

INP et OUT	150
Dossier technique	151

Introduction

Reprise des documentations sur le BASIC du SANYO PCH-25.

La documentation étant peu loquace, le but est d'avoir un document le plus juste et le plus complet possible.

Les tests de syntaxes et autres ont été effectuées pour la plupart sur émulateur.

D'autres ont pu être fait sur des machines réelles.

Le second but de ce document c'est la **GAME JAM 2025** sur **SANYO PHC-25**.

De façon à fournir une bonne base de documentation pour pouvoir créer un logiciel.

Il est toujours possible de participer aux GAME JAM après coup.

C'est d'ailleurs très utile pour compléter les logithèques de ces machines obscures.

La programmation du **Z80** n'est pas abordée ici.

Toutefois, cette documentation essaye d'expliquer le How To pour un programme binaire au travers les instructions pour le langage machine.

La documentation étant quasi nulle sur le sujet.

Pour les **INP** et **OUT** du **SANYO PHC-25** nous aimerais bien avoir plus d'indication.

Les variables

Dans la documentation, elles sont nommées constantes.

Considérant le peu de place en mémoire leur étant réservée, ce n'est pas trop étonnant qu'elles s'appellent constante.

C'est un peu alambiqué, on ne parlera que de variables.

Chaîne de caractères

Elle est constituée par un ou plusieurs caractères enfermés entre guillemets. La forme générale étant "caractère ..."

Exemple :

"PHC 25"

NB : Les guillemets ("") ne sont pas compris dans la chaîne de caractères.

Utilisée pour mémoriser des chaînes de caractères jusqu'à 255 caractères. Les chaînes de caractères utilisent le signe \$ après le nom variable.

Exemple : P\$, DX\$, XI\$, CD\$

Une restriction du BASIC fait qu'il n'est possible d'avoir que 2 caractères pour le nom de la chaîne de caractère.

Exemple :

AB\$
ABC\$

Sont une même variable.

Valeurs

Il y a 3 notations possible, voir ci-après. Il est possible de stocker des nombres de 10^{-39} à 10^{+38} . Il n'y a pas de possibilité de notation binaire.

Notation point (.) fixe

On obtient un nombre négatif ou positif qui inclut le point décimal avec neuf chiffres significatifs.

Exemples :

33.169
-45.765

Notation décimal flottant

On obtient un nombre positif ou négatif exprimé en notation exponentielle. L'exposant va de -38 à +38.

La mantisse peut avoir jusqu'à 9 chiffres significatifs.

Exemple :

123456789 E-38

Notation hexadécimale

La constante hexadécimale est exprimée par une combinaison de caractères de 0 à F précédés par &H.

Exemple :

&HC000

Tableau

Le groupage de données peut être fait sous le terme TABLEAU. La dimension du tableau est définie par la valeur entre parenthèses.

Exemple :

A(3)
K\$(4)

A contient 3 éléments [0,2].

K contient 4 éléments de type chaîne de caractère.

Il est possible d'avoir des tableaux à n dimensions.

Exemple :

X(3,3,3)

Le BASIC

Principe

Les instructions sont mises dans des lignes de code. Ces lignes sont numérotées.

Ce numérotage va de 0 à 65535.

Une ligne numéro 0 est possible, c'est une particularité du BASIC du PHC-25.

Instructions par ligne

Pour mettre Plusieurs instructions sur une ligne de code, on sépare les instructions par « : ».

Tous les espaces peuvent être supprimés dans une ligne de code.

Les instructions

L'ensemble des instructions du PHC-25 sont décrites ci-après.

Il s'agit d'une reprise du document officiel, avec des mises à jours et explications plus précises.
Elles sont rangées par ordre alphabétique.

Les entiers

Il n'y a pas de possibilité d'avoir des entiers comme dans la plupart des autres BASIC.

La notation :

```
10 I%=10
```

N'est pas possible.

Pour adapter un programme qui utilise ce type de variable au PHC-25, il faudra recoder et utiliser l'instruction [INT](#).

NDR : la documentation indique le contraire, c'est à vérifier sur une machine physique.

Les Instructions du BASIC

ABS

Objet

Renvoie la valeur absolue de l'argument fourni.

Syntaxe

ABS (x)

x : Nombre. Peut aussi être une instruction qui renvoie un nombre.

Exemple

```
10 PRINT ABS(9*(-2))
```

Aura comme résultat 18.

Mots clés associés

[DEF FN](#)

AND

Objet

Cette instruction n'est pas documentée.

Provoque un ET logique avec les 2 variables fournies.

Table AND :

A	B	R
0	0	0
0	1	0
1	0	0
1	1	1

Syntaxe

a AND b

a et b doivent être des valeurs numériques.

Exemple

TO DO

Mots clés associés

[IF](#), [NOT](#), [OR](#)

ASC

Objet

Renvoie la valeur du code ASCII du premier caractère d'une chaîne de caractères.

Syntaxe

ASC (a\$)

a\$: Chaîne de caractère.

Exemple

```
10 A$="TEST"  
20 PRINT ASC(A$)
```

Affiche 84 car la valeur ASCII de la lettre T est 84.

Voir le tableau ASCII.

Mots clés associés

[CHR\\$](#), [LEFT\\$](#), [MID\\$](#), [RIGHT\\$](#), [STR\\$](#), [VAL](#)

CHR\$

Objet

Génère un caractère de code ASCII égal à la valeur fournie en argument.
Selon le mode écran, il y a un comportement différent.

Syntaxe

CHR\$ (x)

x : nombre entier de 0 (&h00) à 255 (&hFF).

Exemple

```
PRINT CHR$(42)
PRINT CHR$(20)+CHR$(49)
```

Mots clés associés

[ASC](#), [LEFT\\$](#), [MID\\$](#), [RIGHT\\$](#), [STR\\$](#), [VAL](#)

CLEAR

Objet

Remettre à zéro toutes les variables numériques et à valeur nulle toutes les valeurs alphanumériques.

Syntaxe

CLEAR I,[J]

I : Taille de la mémoire pour y stocker des données. Obligatoire.

J : Optionnel. Limite supérieure de la mémoire pour le programme BASIC.

Exemple

```
CLEAR 400,&hE050
```

400 caractères réservé.

Limite de mémoire supérieur &hE050, adresse au-delà de laquelle on met un programme en langage machine.

Mots clés associés

[FRE](#)

CLOAD

Objet

Charger en mémoire un programme ou un fichier existant sur une cassette.
CLOAD efface le programme précédent avant de charger celui qui est sur la cassette.

Syntaxe

CLOAD"nom"

nom : Le nom est une chaîne dont les six premiers caractères sont significatifs et doivent être identiques au nom sous lequel le programme a été sauvegardé par CSAVE.

Le guillemet de fin n'est pas obligatoire.

CLOAD seul chargera le premier programme de la cassette.

Exemple

CLOAD

Charge le programme depuis le magnétophone.

CLOAD"othelo"

Charge le programme Othelo depuis le magnétophone. Si le programme n'est pas Othelo continu avec le programme suivant sur la cassette.

Mots clés associés

[CLOAD?](#), [CSAVE](#), [SLOAD](#), [SSAVE](#)

CLOAD?

Objet

Une comparaison est faite entre le programme en mémoire et celui qui est sur la cassette.

Syntaxe

`CLOAD?"nom"`

Nom : nom du programme à tester. Seul les 6 premier caractères compte.

Exemple

`CLOAD?"prog"`

Mots clés associés

[CLOAD](#), [CSAVE](#), [SLOAD](#), [SSAVE](#)

CLS

Objet

Tous les caractères affichés sur l'écran sont effacés.

Tous les caractères affichés sur l'écran définie par CONSOLE sont effacés.

Syntaxe

CLS

Exemple

```
10 CLS
```

Efface l'écran.

```
10 CONSOLE 3,5  
20 CLS
```

Seules les lignes 3 à 8 seront effacées.

Les lignes sont numérotées de 0 à 15.

Mots clés associés

[COLOR](#), [CONSOLE](#), [SCREEN](#)

COLOR

Objet

Désignation de la couleur des caractères, ou graphiques affichés sur l'écran.

Même si on désigne les mêmes chiffres, la couleur affichée est différente selon le mode écran choisi (1,2,3,4).

Syntaxe

COLOR a,b,c

a : défini selon les modes la couleur du texte,

b : défini selon les modes 2, 3 et 4, la couleur de fond,

c : défini la palette de couleurs utilisées dans les modes écran 1,2,3 et 4.

Mode 1

Dans ce mode, la couleur de l'écran est désignée par les valeurs a et c sans tenir compte de b.

La couleur de l'écran entier est changée en exécutant l'instruction COLOR et ensuite CLS.

De même, un changement de palette affecte tout l'écran.

Il n'y a pas de graphique possible en mode 1. Ce mode est textuel, de type console.

Les paramètres, a et c peuvent être utilisés seul ou ensemble.

Palettes

Il y a 4 couleurs disponibles dans 2 palettes selon le tableau ci-dessous :

La valeur de c définit la palette utilisée.

Palette 1 :

a	c	Caractère	Fond	Affichage
1	1	Vert Clair	Vert	189 ; 255 ; 181
2	1	Vert	Vert Clair	16 ; 132 ; 8
3	1	Blanc	Orange	255 ; 198 ; 173
4	1	Orange	Blanc	255 ; 66 ; 107

La colonne affichage donne une valeur approchée RGB du texte (ou fond si inversion).

Palette 2 :

a	c	Caractère	Fond	Affichage
1	2	Blanc	Orange	255 ; 198 ; 173
2	2	Orange	Blanc	255 ; 66 ; 107
3	2	Vert Clair	Vert	189 ; 255 ; 181
4	2	Vert	Vert Clair	16 ; 132 ; 8

La colonne affichage donne une valeur approchée RGB du texte (ou fond si inversion).

Exemples :

```
COLOR , ,1
```

Passe la console sur la palette 1, Vert et Vert clair. Le changement de palette affecte tout l'écran.

```
COLOR 1
```

Passe le texte en mode Vert Clair sur fond vert. Affecte toutes les instructions qui suivent jusqu'au prochain COLOR.

COLOR 2

Passe le texte en mode inverse vidéo Vert sur fond Vert Clair. Affecte toutes les instructions qui suivent jusqu'au prochain COLOR.

```
COLOR 1,1,1  
COLOR 1,2,1
```

Ces 2 instruction on le même résultat, le paramètre b est ignoré.

Mode 2

La valeur de a définit la couleur des caractères. Il y a 4 couleurs disponibles. Ce sont les mêmes que le mode 1 ci-dessus.

La valeur de b définit la couleur de fond de l'écran. Il y a 9 couleurs disponibles.

La valeur de c définit la palette de couleur utilisée. Il y a 2 palettes disponibles.

Préalable :

```
SCREEN 2,1,1
```

Ci-dessus, le mode 2 est appliqué à l'écran 1 et est affiché.

NB : Le PHC-25 dispose de la possibilité d'avoir 2 écrans au détriment de l'espace mémoire.

Palettes

Le choix de la palette se fait via la valeur de c.

La valeur de b va peindre le fond, par exemple dans le cas d'un CLS.

Dans le cas des graphiques, fait une inversion entre le noir et la couleur vis-à-vis des fonctions graphiques.

En aucun cas b n'influence l'affichage des caractères.

Palette no 1 :

a	b	c	Caractères	Affichage	Fond	Graphiques
0	0	1	Vert	189 ; 255 ; 181	Noir	Noir
1	1	1	Vert	189 ; 255 ; 181	Vert	Vert
2	2	1	Vert (inversé)	16 ; 132 ; 8	Jaune	Jaune
3	3	1	Orange	255 ; 198 ; 173	Violet	Violet
4	4	1	Orange inversé)	255 ; 66 ; 107	Rouge	Rouge
5	5	1	Orange inversé)		Blanc	Blanc
6	6	1	Orange inversé)		Bleu clair	Bleu clair
7	7	1	Orange inversé)		Rose	Rose
8	8	1	Orange inversé)		Orange	Orange

Exemple :

```
COLOR , ,1
```

Passe sur la palette 1, ceci affecte tout l'écran.

```
10 COLOR ,2  
20 CLS
```

L'écran sera jaune après le CLS.

Palette no 2 :

a	b	c	Caractères	Affichage	Fond	Graphiques
0	0	2	Orange	255 ;198 ;173	Noir	Noir
1	1	2	Orange	255 ;198 ;173	Blanc	Blanc
2	2	2	Orange (inversé)	255 ;66 ;107	Bleu clair	Bleu clair
3	3	2	Vert	189 ; 255 ; 181	Rose	Rose
4	4	2	Vert (inversé)	16 ; 132 ; 8	Orange	Orange
5	5	2	Vert (inversé)		Vert	Vert
6	6	2	Vert (inversé)		Jaune	Jaune
7	7	2	Vert (inversé)		Violet	Violet
8	8	2	Vert (inversé)		Rouge	Rouge

Exemple :

```
COLOR , ,2
```

Passe sur la palette 2, ceci affecte tout l'écran.

Remarque :

Les couleur Rouge et Vert, correspondent au couleur de fond de base des caractères. Il peut être opportun de les utiliser quand il y a besoin de texte à afficher.

Exemple :

```
1000 SCREEN 2,1,1
1010 COLOR , ,1
1020 CLS
1030 PSET (120,120) ,3
1040 LINE(12,12)-(48,48) ,2,BF
1050 LOCATE 5,12:PRINT"Hello World!"
9000 GOSUB 9980
9910 END
9980 K$="" :K$=INKEY$ :IF K$="" THEN 9980
9990 RETURN
```

Mode 3

La valeur de a, définit la couleur des caractères. Il y a 4 couleurs disponibles.

La valeur de b définit la couleur de fond de l'écran. Il y a 5 couleurs disponibles.

La valeur de c définit la palette de couleur utilisée. Il y a 2 palettes disponibles.

Palettes

NDR : La documentation originale est insuffisante.

Mode 4

Ce mode est assez particulier pour définir les couleurs.

Il s'agit d'un mode bicolor avec 4 variantes.

Noter qu'il n'est pas possible de faire de l'affichage texte en inverse.

Palettes

Il y a 4 variantes d'affichage.

Pour choisir la variante, il faut que les valeurs de a et b soient identiques puis fixer c à 1 ou 2 avec l'instruction COLOR. Puis enchaîner avec l'instruction CLS.

Variantes et couleurs :

a	b	c	Caractères	Fond	Graphiques
0	0	1	Vert Clair	Vert	Vert clair
1	1	1	Vert	Vert clair	Vert
0	0	2	Blanc	Noir	Blanc
1	1	2	Noir	Blanc	Noir

Exécuter l'instruction COLOR a,b,c suivie de CLS pour activer la variante.

L'instruction COLOR „c permet de changer de variante. Mais uniquement sur 2 possibilités (a et b = 0 ou a et b = 1).

Exemples

Variante 1.

```
10 SCREEN 4,1,1
20 COLOR 0,0,1:CLS
30 COLOR 1
40 PRINT "Hello World!"
50 LINE (100,100)-(150,150),1,BF
1000 K$=INKEY$:IF K$="" THEN 1000
```

Passe sur la palette 1. Les valeurs a et b étant identique (à 0), met le fond et le texte en Vert lorsque SCREEN s'exécute.

L'instruction COLOR 1 passe le texte en Vert clair.

De même le paramètre de couleur pour les instructions graphiques (0 Vert ; 1 Vert Clair).

Variante 2.

```

10 SCREEN 4,1,1
20 COLOR 1,1,1:CLS
30 COLOR 0
40 PRINT "Hello World!"
50 LINE (100,100)-(150,150),0,BF
1000 K$=INKEY$:IF K$="" THEN 1000

```

Passe sur la palette 1. Les valeurs a et b étant identique (à 1), met le fond et le texte en Vert clair lorsque SCREEN s'exécute.

L'instruction COLOR 0 passe le texte en Vert.

De même le paramètre de couleur pour les instructions graphiques (0 Vert ; 1 Vert Clair).

Variante 3.

```

10 SCREEN 4,1,1
20 COLOR 0,0,2:CLS
30 COLOR 1
40 PRINT "Hello World!"
50 LINE (100,100)-(150,150),1,BF
1000 K$="" :K$=INKEY$:IF K$="" THEN 1000

```

Passe sur la palette 2. Les valeurs a et b étant identique (à 0), met le fond et le texte en Noir lorsque SCREEN s'exécute.

L'instruction COLOR 1 passe le texte en Blanc.

De même le paramètre de couleur pour les instructions graphiques (0 Noir ; 1 Blanc).

Variante 4.

```

10 SCREEN 4,1,1
20 COLOR 1,1,2:CLS
30 COLOR 0
40 PRINT "Hello World!"
50 LINE (100,100)-(150,150),0,BF
1000 K$="" :K$=INKEY$:IF K$="" THEN 1000

```

Passe sur la palette 2. Les valeurs a et b étant identique (à 0), met le fond et le texte en Blanc lorsque SCREEN s'exécute.

L'instruction COLOR 0 passe le texte en Noir.

De même le paramètre de couleur pour les instructions graphiques (0 Noir ; 1 Blanc).

Addendum

Les valeurs a, b, c, peuvent varier de 0 à 255 avec des effets de bord selon le mode choisi.

Mots clés associés

CLS, LOCATE, PRINT, SCREEN

CONSOLE

Objet

Désigne les limites de déplacement vers le bas et le haut de l'écran.
Seules les lignes désignées vont scroller ou être effacées.

Syntaxe

```
CONSOLE a,b
```

a : ligne de départ

b : nombre de ligne (>0) – [1,254]

Exemple

```
10 CLS
20 PRINT "TITRE"
30 CONSOLE 1,15
40 CLS
```

Le titre n'est pas effacé par le CLS de la ligne 40.

```
CONSOLE 0,16
```

Tout l'écran. Depuis la ligne 0 et 16 lignes.

Faire un PRINT sur la dernière ligne définie par CONSOLE entraînera un scroll si le PRINT n'est pas suivi d'un « ; ».

Mots clés associés

[CLS](#), [LOCATE](#)

CONT

Objet

CONT veut dire continuer.

Utilisé pour reprendre l'exécution d'un programme après une suspension par la commande CTRL/C ou après l'exécution d'une instruction STOP.

L'exécution reprend à l'endroit où elle a été interrompue.

CONT n'est plus utilisable si une modification quelconque a été faite dans le programme, ni après une instruction END.

L'écran affiche « Can't continue ».

Syntaxe

CONT

Exemple

CONT

Mots clés associés

[END](#), [STOP](#)

COS

Objet

Le cosinus de X exprimé en radians.
Le résultat est donné en simple précision.

Syntaxe

COS (x)

X : angle en radians.

Exemple

PRINT COS (0.4)

Affichera 0.921060995.

Mots clés associés

[DEF FN](#), [SIN](#), [TAN](#)

CSAVE

Objet

Sauvegarde sur cassette d'un programme en mémoire.

Remarques : Chaque programme sauvegardé est identifié par un nom.

Le BASIC copie le programme existant en mémoire sur la cassette lorsque la commande CSAVE est exécutée. Elle se sert des six premiers caractères pour le « nom ».

Avant l'utilisation d'un CSAVE, il est nécessaire de vérifier que le magnétophone est branché correctement et que les touches RECORD et PLAY sont bien enfoncées.

Syntaxe

```
CSAVE Programme
```

Exemple

```
CSAVE MonProgramme
```

La référence sur la cassette du programme sera « MonPro ». Les 6 premier caractères.

Mots clés associés

[CLOAD](#), [CLOAD ?](#), [SSAVE](#)

CSRLIN

Objet

Obtient la ligne de la position du curseur.

Syntaxe

CSRLIN

Exemple

```
10 LOCATE 12,7  
20 PRINT CSRLIN
```

Le curseur est en position colonne 12, ligne 7, le programme affichera 7.

Mots clés associés

[LPOS](#), [POS](#)

CTOFF

Objet

Fermer l'interrupteur de commande à distance du magnétophone (REMrte Control) arrêtant ainsi le défilement de la cassette.

Sous condition que les connecteurs de prise soient correctement câblés.

Syntaxe

CTOFF

Exemple

Mots clés associés

[CLOAD](#), [CSAVE](#), [CTON](#), [SSAVE](#), [SLOAD](#)

CTON

Objet

Mettre le magnétophone en marche dans le cas où la touche PLAY est enfoncée.
La commande à distance du magnétophone est enclenchée par cette commande, démarrant ainsi la cassette.

Vérifier que la touche PLAY du magnétophone est bien enclenchée.

Sous condition que les connecteurs de prise soient correctement câblés.

Syntaxe

CTON

Exemple

Mots clés associés

[CLOAD](#), [CSAVE](#), [CTOFF](#), [SSAVE](#), [SLOAD](#)

DATA

Objet

Utilisé pour enregistrer dans un programme des valeurs numériques et des caractères constants qui seront appelés par des instructions READ.

Les chaînes de caractères doivent être encadrées de guillemets

Les lignes DATA peuvent se situer n'importe où dans le programme.

Syntaxe

```
DATA valeur[, valeur, ...]
```

Exemple

```
10 FOR I=1 TO 3: READA$(I): NEXT I
20 FOR I=1 TO 3: READB(I): NEXT I
30 FOR I=1 TO 3: PRINT A$(I); "="; B(I); "F": NEXT I
100 DATA "PAIN", "VIN", "JAMBON", 2, 6.50, 10
```

Mots clés associés

[READ](#), [RESTORE](#)

DEF FN

Objet

Définir une fonction écrite par l'utilisateur et lui donner un nom.

NDR : La documentation indique qu'il serait possible d'avoir N paramètre séparé par une virgule.
Apparemment ce n'est pas fonctionnel, il faudrait approfondir le sujet.

Il est possible de se passer des espaces séparateurs.

Syntaxe

DEF FN <nom>(x)=<fonction>

nom : le nom donné à la fonction.

x : le paramètre.

fonction : la formule de la fonction.

Exemples

```
1000 REM
1010 DEF FN SEC(X)=1/COS(X) : REM secante
1020 DEF FN CSC(X)=1/SIN(X) : REM cosecante
1030 DEF FN COT(X)=1/TAN(X) : REM cotangente
1040 DEF FN SINH(X)=(EXP(X)-EXP(-X))/2: REM sinus hyperbolique
1050 DEF FN COSH(X)=(EXP(X)-EXP(-X))/2: REM cosinus hyperbolique
1060 DEF FN TANH(X)=EXP(-X)/(EXP(X)+EXP(-X))*2+1: REM tangente hyperbolique
1070 DEF FN SECH(X)=2/(EXP(X)+EXP(-X)): REM secante hyperbolique
1080 DEF FN CSCH(X)=2/(EXP(X)-EXP(-X)): REM cosecante hyperbolique
1090 DEF FN COTH(X)=EXP(-X)/(EXP(X)-EXP(-X)*2+1: REM cotangente hyperbolique
1100 DEF FN ARCSINH(X)=LOG(X+SQR(X*X+1)): REM inverse sinus hyperbolique
1110 DEF FN ARCCOSH(X)=LOG(X+SQR(X*X-1)): REM inverse cosinus hyperbolique
1120 DEF FN ARCTANH(X)=LOG((1+X)/(1-X))/2: REM inverse tangente hyperbolique
1130 DEF FN ARCSECH(X)=LOG((SQR(-X*X+1)+1/X)): REM inverse secante hyperbolique
1140 DEF FN ARCCSCH(X)=LOG((SGN(X)*SQR(X*X+1))/X): REM inverse cosecante hyperbolique
1150 DEF FN ARCCOTH(X)=LOG((X+1)/(X-1))/2: REM inverse cotangente hyperbolique
```

Défini des fonctions supplémentaires de trigonométrie.

```
1010 DEF FN SEC(X)=1/COS(X)
1010 DEFFNSEC(X)=1/COS(X)
```

Les 2 lignes sont équivalentes.

Exemple :

```
10 DEFNF(X)=3*X+6  
20 PRINT FN F(5)
```

Affichera 21

TO DO test chaînes de caractères.

Mots clés associés

DIM

Objet

Préciser les valeurs maximales des indices pour les tableaux et réserver la place utile en mémoire.
En même temps on assure la place utile pour ranger la mémoire.

Lorsqu'un tableau de variables est utilisé sans instruction DIM, la valeur maximum est considérée comme étant 10 lignes en 3 dimensions.

Une erreur apparaît si l'on fait appel à une valeur plus grande.

La valeur minimale est zéro.

Syntaxe

```
DIM variable[,variable, ...]
```

Exemple

```
10 DIM B$(20,10), C(30), E(10,3,5,7)
```

Mots clés associés

ELSE

Suivant les résultats d'un test, l'on prendra une décision ou l'on choisira une suite pour l'exécution du programme.

Si le résultat du test est vrai, c'est-à-dire différent de zéro, alors l'instruction THEN suivra jusqu'à ELSE ou la fin de la ligne et sera exécutée.

Si l'expression est égale à zéro (valeur fausse), seules les instructions suivant ELSE, si elles existent, seront exécutées.

Syntaxe

```
IF <expression> THEN <instruction> ELSE <instruction>
```

Expression : Expression à tester

Instruction : toutes instructions valides du BASIC, si il y en a plusieurs, les séparer par “:”.

Exemple

```
5 A=-5: B=-10
10 IF A<>0 THEN B=10 ELSE B=0: GOTO 100
30 PRINT A,B
40 END
100 PRINT A,B
110 A=A+1
120 GOTO 10
```

Mots clés associés

AND, IF, NOT, THEN, OR

END

Objet

Terminer l'exécution d'un programme.

L'instruction END peut être placée n'importe où dans le programme pour terminer l'exécution.

Le message BREAK n'est pas imprimé lorsque l'instruction END est exécutée.

La commande CONT ne peut être utilisée après un END.

Syntaxe

END

Exemple

9990 END

Le programme s'arrête lorsqu'il atteint la ligne 9990.

Mots clés associés

BREAK, CONT

EXEC

Objet

Le programme en basic ira à l'adresse indiquée par la variable X pour exécuter un programme écrit en langage machine (hexadécimal). Le programme aura été écrit grâce à l'instruction POKE et devra se finir automatiquement par la valeur (&HC9) (RET) qui permettra un retour au langage Basic.

Syntaxe

EXEC &Hx

X est l'adresse hexadécimale du programme à exécuter.

Exemple

```
10 CLEAR 100,&HF000
20 LET J=00
30 INPUT "valeur Hex";A$
40 N=VAL("&H"+A$)
50 IF A$="ZZ" OR A$="zz" THEN 90
60 POKE &HF000+(J),N
70 J=J+1
80 GOTO 30
90 END
100 CLS
110 EXEC &HF000
120 PRINT "SORTIE"
130 END
```

Valeur à passer : 3E, 41, 32, 8F, 60, C9

Le programme met le caractère A à la position 16,5 de l'écran (LOCATE). Pour les modes SCREEN 1 et SCREEN 2.

Son adresse mémoire étant &h608F.

Voir chapitre sur l'écran du SANYO PHC-25 pour plus d'information.

Mots clés associés

[PEEK](#), [POKE](#), [USR](#)

EXP

Objet

Renvoie la valeur de e à la puissance x.

Syntaxe

EXP (x)

x : Puissance pour e.

Exemple

10 PRINT EXP(6)

Retourne : 403.428794.

Mots clés associés

[DEF FN](#)

FRE

Objet

Retourne la valeur de la mémoire disponible.

Syntaxe

FRE (x)

x :

FRE (x\$)

x\$:

Exemple

10 PRINT FRE(x)

Nombre d'octets restant dans la mémoire pour le BASIC.

10 PRINT FRE(x\$)

Nombre d'octets restant dans la mémoire pour les chaînes de caractères.

Mots clés associés

CLEAR

FOR

Objet

FOR NEXT répète et exécute une chaîne de commandes le nombre de fois désigné. La première expression X, est la valeur initiale, Y est une valeur limite à ne pas dépasser. Les instructions entre FOR et NEXT sont exécutées autant de fois que nécessaire avec chaque fois incrémentation du compteur variable de la quantité indiquée après l'instruction STEP pour que le compteur dépasse la limite Y. Dans un tel cas, l'exécution passe à l'instruction suivante NEXT.

Syntaxe

```
FOR variable=X TO Y (STEP Z)
...
NEXT
```

NEXT est obligatoire.

Exemple

Mots clés associés

[NEXT](#), [STEP](#), [TO](#)

GOSUB

Objet

Aller à un sous-programme et revenir.

Syntaxe

GOSUB adresse

adresse : numéro de ligne du sous-programme.

Exemple

```
1010 PRINT"BCD"
1020 GOSUB 1050
1030 PRINT"DEF"
1040 END
1050 PRINT"123"
1060 RETURN
```

Mots clés associés

[GOTO](#), [ON](#), [RETURN](#)

GOTO

Objet

Saut inconditionnel à une ligne qui a été désignée.
Une erreur est signalée si la ligne désignée n'existe pas.

Syntaxe

```
GOTO adresse
```

adresse : ligne cible du saut inconditionnel.

Exemple

Mots clés associés

GOSUB, ON

IF

Objet

Suivant les résultats d'un test, l'on prendra une décision ou l'on choisira une suite pour l'exécution du programme.

Si le résultat du test est vrai, c'est-à-dire différent de zéro, alors l'instruction THEN suivra jusqu'à ELSE ou la fin de la ligne et sera exécutée.

Si l'expression est égale à zéro (valeur fausse), seules les instructions suivant ELSE, si elles existent, seront exécutées.

Syntaxe

```
IF <expression> THEN <instruction> ELSE <instruction>
```

THEN est obligatoire. Si expression est VRAI alors les instructions sont exécutées.

ELSE est optionnel. Si expression est FAUSSE alors les instructions sont exécutées.

THEN et ELSE doivent être sur la m^{ême} ligne de code.

Exemple

```
5 A=-5: B=-10
10 IF A<>0 THEN B=10 ELSE B=0: GOTO 100
30 PRINT A,B
40 END
100 PRINT A,B
110 A=A+1
120 GOTO 10
```

Mots clés associés

AND, ELSE, NOT, OR, THEN

INKEY\$

Objet

Obtient le caractère de la touche lorsque l'on appuie sur le clavier.
Permet de tester quelle touche a été appuyée.

Syntaxe

INKEY\$

Exemple

```
100 IF INKEY$="" THEN GOTO 100
110 IF INKEY$<>"A" THEN GOTO 100
```

Ligne 100, tant qu'aucune touche n'est appuyée retourne à la ligne 100
Ligne 110, test si la touche A majuscule a été appuyée.

Voir aussi [Trucs et Astuces](#).

Mots clés associés

[STICK](#), [STRIG](#)

INP

Objet

Lecture d'un octet sur le port I (adresse de &H00 à &HFF).
Voir aussi le chapitre [INP et OUT](#).

Syntaxe

`INP (x)`

x : adresse du port de 0 à 255.

Exemple

`PRINT INP(&H8F)`

Renvoie la valeur du port &H8F.

Mots clés associés

[OUT](#)

INPUT

Objet

Lorsqu'une instruction INPUT est exécutée, le point d'interrogation est affiché, indiquant une demande de l'entrée des données. S'il y a une « chaîne de caractères », elle est affichée devant le point d'interrogation(?).

NDR : La fonction INPUT provoque un retour en SCREEN 1 lors de son utilisation en SCREEN 3 et 4.

Syntaxe

```
INPUT [chaine,]variable[,variable]
```

Variable peut être une chaîne de caractère ou une variable numérique.

Exemple

```
INPUT A
INPUT A$
INPUT »Donner votre valeur »,A
```

Mots clés associés

INPUT#

INPUT#

Objet

Lecture de données dans un fichier séquentiel et affectation des valeurs aux variables de la liste.
Le numéro du fichier est le numéro associé au fichier au moment de son ouverture.

La liste contient le nom des variables où l'on doit affecter les données existantes sur le fichier.
Il faut que les types concordent et il n'y a pas de point d'interrogation sur l'écran comme dans le « INPUT ».

Lorsque l'on entre des données par l'intermédiaire du magnétophone, il est nécessaire
d'enregistrer ces données par une instruction [PRINT#](#).

Syntaxe

`INPUT#-n, val`

n :

0 : clavier

1 : magnétophone

Val : élément à lire, la donnée enregistrée doit correspondre au type renseigné.

Exemple

Mots clés associés

INPUT, PRINT, PRINT#

INT

Objet

Retourne le plus grand entier inférieur ou égal à X.

Syntaxe

`INT(x)`

x : nombre dont on veut l'entier.

Exemple

```
PRINT INT(99.99)
```

Renvoie 99.

```
PRINT INT(-32,12)
```

Renvoie -33, c'est bien la valeur inférieur du nombre passé en argument.

Mots clés associés

[DEF FN](#)

KEY

Objet

Les touches F1, F2, F3 et F4 sont reprogrammables en mode normal et en mode SHIFT. Ce qui donne 8 touches au total.

Une touche peut contenir 8 caractères exécutables au maximum.

Syntaxe

```
KEYn, char
```

Exemple

```
KEY1, "CONSOLE"
```

La touche F1 écrira « CONSOLE ».

```
KEY7, "CTON"+CHR$(13)
```

La touche F3 (avec SHIFT) mettra le moteur du magnétophone en marche.

Mots clés associés

LCOPY

Objet

Envoi de l'information contenue sur l'écran vers l'imprimante. Aussi appelée « HARD COPY ». Ne marche que sur imprimante graphique.

NDR : fonction à tester dans un programme.

Syntaxe

LCOPY

Exemple

Mots clés associés

[LPRINT](#), PRINT#

LEFT\$

Objet

Sélectionne le nombre spécifié de caractère à gauche de la chaîne fourni en argument.

Syntaxe

LEFT\$ (A\$, x)

A\$: Chaîne de caractère sur laquelle on veut faire l'extraction.

x : nombre de caractère à extraire.

Si x vaut 0, renvoie une chaîne vide.

La valeur de x ne doit pas dépasser 255.

Si la chaîne est plus courte que le x spécifié, LEFT\$ renvoie toute la chaîne.

Exemple

Mots clés associés

[ASC](#), [CHR\\$](#), [DEF FN](#), [MID\\$](#), [RIGHT\\$](#), [STR\\$](#), [VAL](#)

LEN

Objet

Renvoie la longueur de la chaîne fournie en argument.

Syntaxe

LEN (A\$)

A\$: Chaîne dont on veut connaître la longueur

Exemple

```
10 D$ = "LES QUATRE SAISONS"
20 PRINT LEN(D$)
```

Mots clés associés

LET

Objet

La valeur de "expression" est substituée et devient une variable.

En fait, le signe égal (=) suffit pour constituer l'affectation de la variable.

L'instruction LET a le même but.

NDR : Vous devez considérer cette instruction comme obsolète.

Dans le cadre d'une adaptation de programme, supprimer l'instruction LET pour gagner de la mémoire.

Syntaxe

```
LET NomVariable = Expression
```

Exemple

```
10 LET A=B+1
```

```
10 A=B+1
```

Les 2 lignes sont identiques sur le PHC-25.

Mots clés associés

LINE

Objet

Relier les points par une ligne ou les insérer dans un rectangle.

Deux points désignés par des références données sont reliés par une ligne qui a une couleur désignée.

Lorsque "B" est désigné, un rectangle est dessiné utilisant les deux points comme angles opposés.
De plus, si F est désigné, le rectangle est coloré par la couleur spécifiée ou celle de la dernière instruction COLOR.

Les définitions en ligne et en colonne sont assujetties à la définition du SCREEN (1,2, 3 ,4).

Syntaxe

```
LINE (X1,Y1)-(X2,Y2), (couleur), (BF)
```

X1 : coordonnées sur l'axe horizontale

Y1 : coordonnées sur l'axe verticale

X2 : coordonnées sur l'axe horizontale

Y2 : coordonnées sur l'axe verticale

Couleur :

B : Bordure, il faut utiliser la lettre « B »

F : Fill (remplissage), il faut utiliser la lettre « F »

Exemple

```
LINE (5,5)-(20,20), 3
```

Les points de coordonnées 5,5 et 20,20 sont reliés par une ligne de la couleur 3

```
LINE (5,5)-(20,20), 3,B
```

Les points 5,5 et 20 ,20 appartiennent à un rectangle. Ils sont les deux points opposés.

Exemple : LINE(10,8)-(190,130),2,BF

Un rectangle colorié de couleur 2 passe par les points de coordonnées 10-8 et 190-130

Mots clés associés

COLOR, PAINT, PRESET, PSET, SCREEN

LIST

Objet

Cette commande affiche sur l'écran le programme en mémoire.

Syntaxe

LIST [n-m]

Si n n'est pas spécifié, affiche tout le programme.

Si n est spécifié, affiche la ligne n.

Si n-m sont spécifiés, affiche les lignes n à m.

Si -m est spécifié, affiche du début du programme jusqu'à la ligne m.

Exemple

LIST

Affiche tout le programme.

LIST 1000

Affiche la ligne 1000.

LIST 100-150

Affiche toutes les lignes de 100 à 150.

LIST 100-

Affiche toutes les lignes depuis la ligne 100 jusqu'à la fin du programme.

NDR : à tester à l'intérieur d'un programme.

Mots clés associés

[LLIST](#)

LLIST

Objet

Lister sur l'imprimante tout ou partie du programme existant en mémoire.
Assume le fait que l'imprimante fait 80 caractères de large.

Syntaxe

LLIST [n-m]

Si n n'est pas spécifié, affiche tout le programme.
Si n est spécifié, affiche la ligne n.
Si n-m sont spécifiés, affiche les ligne n à m.
Si -m est spécifié, affiche du début du programme jusqu'à la ligne m.

Exemple

LLIST

Imprime tout le programme.

LLIST 1000

Imprime la ligne 1000.

LLIST 100-150

Imprime toutes les lignes de 100 à 150.

LLIST 100-

Imprime toutes les lignes depuis la ligne 100 jusqu'à la fin du programme.

NDR : à tester à l'intérieur d'un programme.

Mots clés associés

[LIST](#)

LOCATE

Objet

Le curseur est déplacé vers une position sur l'écran qui est exprimée par une position horizontale et verticale.

Le coin gauche en haut de l'écran est 0,0.

La position horizontale (colonne) va de 0 à 31 pour les modes 1,2,4 d'écran.

Toutefois, lorsque l'écran 3 est utilisé, la position horizontale s'arrête à 15.

La position verticale (ligne) va de 0 à 15 pour tous les modes d'écran.

LOCATE n'est pas affecté par CONSOLE. Par contre l'inverse est vrai.

Syntaxe

```
LOCATE X,Y
```

X : Colonne, [0,31] ou [0,16] en SCREEN 3.

Y : Ligne, [0,16].

Exemple

```
10 FOR I=1 TO 100
20 LOCATE 10,10
30 PRINT I
40 NEXT
```

Mots clés associés

[CONSOLE](#)

LOG

Objet

Renvoie le logarithme népérien de l'argument. Cet argument doit être positif.

Syntaxe

`LOG(x)`

X doit être positif.

Exemple

`PRINT LOG(5.4)`

Mots clés associés

[DEF FN](#)

LPOS

Objet

Renvoie la position de tête du pointeur de ligne dans le tampon de sortie de l'imprimante.

Syntaxe

LPOS (X)

Exemple

```
10 LPRINT "DIMANCHE"  
20 PRINT LPOS (X)
```

Aura pour résultat 8.

Mots clés associés

[CSRLIN](#), [POS](#)

LPRINT

Objet

Écriture de données sur l'imprimante.

Cette instruction est identique à PRINT, la sortie se fait sur l'imprimante connectée au PHC 25.

Syntaxe

```
LPRINT val
```

Val est une valeur soit alphanumérique soit numérique.

Exemple

```
LPRINT "Bonjour. "
```

Mots clés associés

[LCOPY](#), [LLIST](#), [LPOS](#), [LPRINT](#), [PRINT#](#)

MID\$

Objet

Renvoie une sous chaîne de la longueur spécifiée depuis un rang spécifié.

Syntaxe

MID\$ (A\$, R, L)

R : Rang de début d'extraction.

L : Nombre de caractère à extraire.

R et L doivent être compris entre 0 et 255.

Exemple

```
10 A$= "BONJOUR"
20 PRINT MID$(A$,2,4)
```

Mots clés associés

[ASC](#), [CHR\\$](#), [DEF FN](#), [LEFT\\$](#), [RIGHT\\$](#), [STR\\$](#), [VAL](#)

NEW

Objet

Effacer le programme présent en mémoire et remet à zéro toutes les variables.

Syntaxe

NEW

Exemple

NEW

Mots clés associés

CONT, END, STOP

NEXT

Objet

Syntaxe

Exemple

Mots clés associés

FOR, STEP

NOT

Objet

Cette instruction n'est pas documentée.

Provoque un NOT logique avec la variable fournie.

Table :

A	R
0	1
1	0

Syntaxe

NOT a

a doit être une valeur numérique.

Exemple

TO DO

Mots clés associés

[AND](#), [IF](#), [OR](#)

ON GOSUB

Objet

Branchement multiple suivant la valeur de l'expression. C'est cette valeur qui détermine sur quelle ligne le programme sera branché.

Si la valeur est I, le programme sera branché sur I.

Si l'expression n'est pas intégrale (entière) les décimales seront arrondies.

Si la valeur de l'expression est zéro ou négative ou si elle est plus grande que les numéros de ligne, le programme passera à la ligne suivante.

Syntaxe

ON Valeur GOSUB L1,L2,Ln...

Valeur est la valeur sur laquelle doit s'effectué le test.

En fonction du résultat, branchement sur la ligne n.

Exemple

```
ON K GOSUB 100,200,300,400
```

Si K<=0, ligne suivante

Si K>=5, ligne suivante

Si K=1, ligne 100

Si K=2, ligne 200

Si K=3, ligne 300

Si K=5, ligne 400

Après le RETURN, ligne suivante.

Mots clés associés

GOSUB, GOTO, RETURN

ON GOTO

Objet

Branchement multiple suivant la valeur de l'expression. C'est cette valeur qui détermine sur quelle ligne le programme sera branché.

Si la valeur est I, le programme sera branché sur I.

Si l'expression n'est pas intégrale (entière) les décimales seront arrondies.

Si la valeur de l'expression est zéro ou négative ou si elle est plus grande que les numéros de ligne, le programme passera à la ligne suivante.

Syntaxe

ON Valeur GOTO L1,L2,Ln...

Valeur est la valeur sur laquelle doit s'effectué le test.

En fonction du résultat, branchement sur la ligne n.

Exemple

ON K GOTO 100,200,300,400

Si K<=0, ligne suivante

Si K>=5, ligne suivante

Si K=1, ligne 100

Si K=2, ligne 200

Si K=3, ligne 300

Si K=5, ligne 400

Mots clés associés

GOSUB, GOTO, RETURN

OR

Objet

Cette instruction n'est pas documentée.

Provoque un OU logique avec les 2 variables fournies.

A	B	R
0	0	0
0	1	1
1	0	1
1	1	1

Syntaxe

a OR b

a et b doivent être des valeurs numériques.

Exemple

TO DO

Mots clés associés

[AND](#), [IF](#), [NOT](#)

OUT

Objet

Cette instruction permet d'envoyer sur le port I l'octet J.

NDR : nécessite plus d'investigation pour connaître tous les INP et OUT du PHC-25.

Syntaxe

```
OUT I,J
```

I : No de port,

J : Octet à envoyer.

Exemple

```
OUT &h20,&h64
```

La valeur &h64 sera envoyée sur le port no &h20.

Mots clés associés

[INP](#)

PAINT

Objet

Les coordonnées x et y définissent 1 point se situant dans une zone de l'écran. Cette zone est peinte avec la couleur C sauf le périmètre de couleur P.

Syntaxe

```
PAINT (X,Y),c,p
```

X : coordonnée horizontale.

Y : coordonnée verticale.

c : couleur de remplissage selon le SCREEN.

p : couleur de périmètre selon le SCREEN.

Si p est omis ou si p est égale à c alors le périmètre aura la couleur définie par c.

Exemple

```
10 LINE(0,0)-(255,0),2,BF  
20 LINE(255,0)-(255,191),2,BF  
30 LINE (255,191)-(5,186),2,BF  
40 LINE(5,186)-(0,0,),2,BF  
50 PAINT (80,80),4,2
```

NDR : exemple à revoir car nécessite SCREEN.

Mots clés associés

COLOR, LINE, PRESET, PSET, SCREEN

PEEK

Objet

Renvoie la valeur de l'octet de l'adresse spécifiée. Les valeurs retournées vont de &H00 à &HFF.

Syntaxe

PEEK(x)

X : adresse mémoire dont on veut lire l'octet.

Exemple

```
A=PEEK(&H2A00)
```

Mots clés associés

EXEC, POKE, USR

PLAY

Objet

La musique est produite par le générateur de son (vendu en option).

Syntaxe

PLAY"ox [sx] [mx] [vx] [lx] [tx] [rx] [x+] [x-] n"

Les fonctions entre crochet sont facultatives

Avec n: (cdefgab), notation Anglo-saxonne de la musique.

c	d	e	f	g	a	b
do	ré	mi	fa	sol	la	si

Dans l'instruction PLAY, pour la chaîne de caractères, les caractères suivants sont utilisés pour donner des significations spéciales :

ox	Désigne l'octave (l'octave plus haute est o4) initial 4
sx	Désigne la forme de l'enveloppe (se référer à l'annexe D.)
mx	Désigne la période de l'enveloppe (1<=x<=65535)
vx	Désigne le volume (0<=x<=15) initial 8
lx	Désigne la durée du son (0<=x<=64)
tx	Désigne la vitesse (tempo) du son initial 170
rx	Désigne la durée de la période de silence (aucun son) (1<=x<=64)
x+	Le son est remonté d'un demi-ton
x-	Le son est descendu d'un demi-ton

Tous les caractères de PLAY doivent être en minuscule.

Exemple

PLAY"o4 18 v2 s5 a"

octave 4 ; longueur 8 ; volume 2 ; enveloppe 5 voix ; note LA

```
10 FOR I=0 TO 10
20 PLAY"o6132c"
30 NEXT I

10 FOR I=0 TO 10
20 PLAY"o41cdfgab05c"
30 NEXT I

10 PLAY
"o7fardl34fafdbcfa05farecferafbcl56gbfddeaedaeaddcffdfbfebacdfeadaeo4129ggabgag
afdfdfaffaddeeaddbccffcfdo7ffacca"
```

Mots clés associés

SOUND

POINT

Objet

Renvoie le numéro de la couleur sélectionnée par les coordonnées (x,y).

Syntaxe

POINT (x, y)

Exemple

```
10 PSET(7,30),3  
20 PRINT POINT(7,30)
```

Mots clés associés

PRESET, PSET

POKE

Objet

Insert un octet à une adresse désignée de la mémoire.

Syntaxe

```
POKE Expression 1, Expression 2
```

Expression 1 : Adresse mémoire

Expression 2 : Octet à placer (&H00 à &HFF)

Les valeurs peuvent être sous forme hexadécimale ou décimale.

Exemple

```
10 POKE &H6010,&H81
20 POKE 4053,24
```

Ligne 10 : Affiche le caractère pour les mode SCREEN 1 et SCREEN 2.

Ligne 20 : place à l'adresse &h0FD5 l'octet de valeur &H18.

```
POKE &h6080,42
```

Affiche le caractère « * » première colonne 5^{ème} ligne pour les modes SCREEN 1 et SCREEN 2.

Mots clés associés

[EXEC](#), [PEEK](#), [USR](#)

POS

Objet

Renvoie la position du curseur sur la colonne.

Syntaxe

`POS (X)`

Exemple

```
10 LOCATE 12,7  
20 PRINT POS(X)
```

Afficher 12.

Mots clés associés

[CSRLIN](#), [LPOS](#)

PRESET

Objet

Effacer le point de coordonnées (X,Y) sur l'écran.

Forte dépendance vis-à-vis de l'instruction SCREEN.

Syntaxe

```
PRESET (X,Y)
```

X : coordonnée horizontale.

Y : coordonnée vertical.

Exemple

```
5 SCREEN 4,1,1 TO 20
6 CLS
9 FOR X=1 TO 100
10 PSET (X,100),3
11 IF X> 20 THEN PRESET (X,100)
12 IF X> 50 THEN PSET(X,100),3
15 NEXT
20 GOTO 20
```

Mots clés associés

COLOR, LINE, PAINT, PRESET, PSET, SCREEN

PRINT

Objet

Écrire des données sur l'écran.

Si la liste d'expressions est absente, une ligne blanche s'en suivra.

La position de chaque élément de la ligne est déterminée par la ponctuation.

Syntaxe

```
PRINT élément[,|;[élément]]...
```

Les caractères doivent obligatoirement être encadrés entre guillemets.

Lorsque l'on écrit un nombre multiple d'expressions, chacune doit être séparée par une virgule (,) ou un point-virgule (;).

Une virgule entre deux expressions de la liste fait que l'expression suivant la virgule est imprimée au début de la zone suivante.

Si c'est un point-virgule, l'expression se fait immédiatement après la dernière expression écrite.

Un ou plusieurs espaces entre deux expressions ont le même effet que le point-virgule.

Exemple

```
10 A$="PROGRAMME"
20 B$="No.":C=15
30 D$="exemple de programme"
40 PRINT D$
50 PRINT
60 PRINT A$;B$;C
70 PRINT "suivant"

PRINT"Hello World !
```

Cette instruction fonctionne, malgré l'absence du guillemet en fin de ligne.

Va afficher tous les caractères présents jusqu'à la fin de la ligne.

Mots clés associés

INPUT, INPUT #, LOCATE, LPRINT, PRINT#, SPC, TAB

PRINT#

Objet

Écriture des données dans un appareil désigné.

La désignation de l'appareil se fait par le numéro suivant PRINT #

Syntaxe

PRINT#-n

N=0 : écran

N=1 : magnétophone

N=3 : imprimante.

Exemple

Mots clés associés

INPUT, INPUT#, LOCATE, LPRINT, PRINT, SPC, TAB

PSET

Objet

Un point ayant une couleur désignée est affiché à l'endroit spécifié par les coordonnées X,Y.
La couleur c dépend du SCREEN 1,2,3,4 et de la couleur choisie (COLOR).

Syntaxe

```
PSET(X, Y) ,c
```

X : coordonnée horizontale

Y : coordonnée verticale

C : couleur

Exemple

Mots clés associés

COLOR, LINE, PRESET, PSET, SCREEN

READ

Objet

Lecture des valeurs dans une instruction DATA et affectation aux variables citées. Une instruction READ doit toujours être utilisée en relation avec une ou plusieurs instructions DATA.

L'instruction READ distribue les valeurs données dans l'instruction DATA aux variables mentionnées.

Ces variables sont numériques ou de type chaîne et doivent s'accorder, sinon il peut y avoir "erreur de syntaxe".

S'il reste des DATA inutilisés, le READ suivant les utilise.

S'il n'en reste pas, ils seront ignorés.

On peut relire les DATA grâce à l'instruction RESTORE.

Syntaxe

```
READ valeur[,valeur]
```

Valeur est de type numérique ou alphanumérique.

Exemple

```
READ A$  
READ A  
READ A,A$
```

Mots clés associés

[DATA](#), [RESTORE](#)

REM

Objet

Insérer des remarques et des notes dans le programme.

Bien que l'instruction REM ne soit pas exécutable, elle figure dans la liste avec le contenu du texte qui l'accompagne.

Les instructions qui suivraient REM après un deux-points « : » ne sont pas exécutées.

Syntaxe

```
REM [texte]
```

Texte est optionnel.

Exemple

```
10 REM Menu
```

```
10 REM Menu: PRINT
```

L'instruction PRINT ne sera pas exécutée.

Mots clés associés

RESTORE

Objet

Permettre de ·relire des données à partir d'une certaine ligne.

Après l'instruction RESTORE, le premier READ prend ses valeurs dans la première instruction DATA.

Lorsque le numéro de ligne est spécifié, la lecture se fera à partir de cette ligne.

Syntaxe

RESTORE [n]

N : numéro de ligne pour relire les données.

Exemple

RESTORE

Restaure la lecture au premier DATA du programme.

RESTORE 100

Restaure la lecture à la ligne 100, ou au premier DATA après la ligne 100.

Mots clés associés

[DATA](#), [READ](#)

RETURN

Objet

Les instructions qui suivraient RETURN après un deux-points « : » ne sont pas exécutées.

Syntaxe

RETURN

Exemple

...
1050 RETURN

Mots clés associés

[GOSUB](#), [ON GOSUB](#)

RIGHT\$

Objet

Sélectionne le nombre spécifié de caractère à droite de la chaîne fourni en argument.

Syntaxe

RIGHT\$ (A\$, n)

A\$: Chaîne de caractère sur laquelle on veut faire l'extraction.

n : nombre de caractère à extraire.

Si x vaut 0, renvoie une chaîne vide.

La valeur de x ne doit pas dépasser 255.

Si la chaîne est plus courte que le x spécifié, RIGHT\$ renvoie toute la chaîne.

Exemple

```
10 A$="ABCDEFG"  
20 PRINT RIGHT$(A$, 4)
```

Mots clés associés

[ASC](#), [CHR\\$](#), [DEF FN](#), [LEFT\\$](#), [MID\\$](#), [STR\\$](#), [VAL](#)

RND

Objet

Initialiser le générateur de nombres aléatoires.

Si l'expression facultative manque, l'exécution est suspendue.

Si l'on utilise RND sans initialiser le générateur à chaque exécution du programme, la même suite de nombres aléatoires sera produite.

Syntaxe

RND (x)

x>0 commence une nouvelle séquence

x=0 donne le dernier nombre généré

x<0 génère un nouveau nombre aléatoire

Exemple

```
10 FOR I=1 TO 5
20 PRINT INT(RND(1)*100)
30 NEXT I

10 CLS:PRINT"RND TEST"
20 A=1:B=13
30 I=RND(-TIME):PRINT"SEED:",I: REM NEW SEED
40 FOR I=1 TO 10
50 N=A+INT((B-A+1)*RND(1)): REM BETWEEN [A,B]
60 PRINT "E";I;" :";A;" -"; B; " ="; N
70 NEXT I
80 END
```

Crée une nouvelle série et génère un nombre entre A et B.

Mots clés associés

[DEF FN](#)

RUN

Objet

Lancement de l'exécution du programme stocké en mémoire.
La touche F1 contient par défaut l'instruction RUN.

Syntaxe

RUN [n]

N : numéro de ligne, n'est pas obligatoire.

Exemple

RUN

Mots clés associés

[END](#), [STOP](#)

SCREEN

Objet

Le PHC 25 a deux pages d'écran qui sont définies à la mise sous tension.

- mode avec 1 page d'écran, la mémoire disponible sera de 14265 octets.
- mode avec 2 pages d'écran, la mémoire disponible sera de 8121 octets.

Ce qui correspond à la question posée à l'allumage « SCREEN ? ».

Il ne faut pas confondre le choix du nombre de pages utilisées et la façon d'utiliser ces pages.

L'instruction SCREEN provoque l'équivalent d'un [CLS](#).

Voir aussi l'instruction [COLOR](#).

Syntaxe

SCREEN a,b,c

La valeur a fixe le choix de la résolution d'écran.

Il y a 4 possibilités d'utilisation d'une page suivant le tableau ci-dessous.

a	1	2	3	4
Texte	16L 32C	16L 32C	16L 16C	16L 32C
Graphique	(16x32)*	64x48	128x192	256x192

(*) Les instructions graphiques fonctionnent.

La valeur b désigne le numéro de la page affectée par les instructions qui suivent.

La valeur c désigne le numéro de la page affichée à l'écran. (NDR : selon le choix fait au démarrage de la machine).

Les variables b et c ne sont utilisables qu'avec 2 pages écran (ce qui correspond au 2 que l'on a choisi à la mise sous tension).

Exemple

Voir ci-après pour les 4 mode d'écran.

Mots clés associés

[CLS](#), [COLOR](#), [CONSOLE](#)

SCREEN 1

Objet

Mode caractères.

Voir information avec [SCREEN](#).

Voir aussi l'instruction [COLOR](#).

Syntaxe

```
SCREEN 1,b,c
```

La valeur b désigne le numéro de la page affectée par les instructions qui suivent.

La valeur c désigne le numéro de la page affichée à l'écran. (NDR : selon le choix fait au démarrage de la machine).

Les valeurs b et c n'ont pas de signification si 1 selon écran à été demandé au démarrage de la machine.

Exemple

Mode 1 écran :

```
SCREEN 1
```

Mode 2 écrans :

```
SCREEN 1,1,1
```

Mots clés associés

[CLS](#), [COLOR](#), [CONSOLE](#)

SCREEN 2

Objet

Mode caractère et semi-graphique.

Voir information avec [SCREEN](#).

Voir aussi l'instruction [COLOR](#).

Syntaxe

```
SCREEN 2,b,c
```

La valeur b désigne le numéro de la page affectée par les instructions qui suivent.

La valeur c désigne le numéro de la page affichée à l'écran. (NDR : selon le choix fait au démarrage de la machine).

Exemple

Mode 1 écran :

```
SCREEN 2
```

Mode 2 écrans :

```
SCREEN 2,1,1
```

Mots clés associés

[CLS](#), [COLOR](#), [CONSOLE](#)

SCREEN 3

Objet

Mode graphique.

Voir information avec [SCREEN](#).

Voir aussi l'instruction [COLOR](#).

Syntaxe

SCREEN 3,b,c

La valeur b désigne le numéro de la page affectée par les instructions qui suivent.

La valeur c désigne le numéro de la page affichée à l'écran. (NDR : selon le choix fait au démarrage de la machine).

Exemple

Mode 1 écran :

SCREEN 3

Mode 2 écrans :

SCREEN 3,1,1

Mots clés associés

[CLS](#), [COLOR](#), [CONSOLE](#)

SCREEN 4

Objet

Mode graphique.

Voir information avec [SCREEN](#).

Voir aussi l'instruction [COLOR](#).

Syntaxe

SCREEN 4,b,c

La valeur b désigne le numéro de la page affectée par les instructions qui suivent.

La valeur c désigne le numéro de la page affichée à l'écran. (NDR : selon le choix fait au démarrage de la machine).

Exemple

Mode 1 écran :

SCREEN 4

Mode 2 écrans :

SCREEN 4,1,1

Mots clés associés

[CLS](#), [COLOR](#), [CONSOLE](#)

SCRIN

Objet

Renvoie la valeur ASCII du caractère de coordonnées d'écran (c,l).

Syntaxe

```
SCRIN(c,l)
```

c : numéro de la colonne

L : numéro de la ligne

Exemple

```
5 CLS
10 PRINT"ABCDEF"
20 Z=SCRIN(5,0)
30 PRINT Z
```

La valeur renvoyée est bien 70 qui est le code ASCII de F.

Les coordonnées texte commence en haut à gauche (0,0).

Mots clés associés

CHR\$, LOCATE

SGN

Objet

Connaître le signe d'un nombre.

Syntaxe

SGN (x)

x : Nombre dont on veut connaître le signe.

Renvoie 1 si $x \geq 0$

Renvoie -1 si $x < 0$

Exemple

```
PRINT SGN(912)
```

Affichera 1.

Mots clés associés

[DEF FN](#)

SIN

Objet

Calcule le sinus de l'angle en radians. Le résultat est obtenu en précision simple.

Syntaxe

SIN(x)

x : angle dont on veut le sinus.

Exemple

```
PRINT SIN(0.32)
```

Résultat : .314566561

Mots clés associés

COS, [DEF FN](#), TAN

SLOAD

Objet

L'information de l'écran mémorisée par la cassette est enregistrée dans la mémoire de l'écran.
Si la page de l'écran utilisée n'est pas celle qui a été mémorisée, la commande SLOAD est arrêtée et une ERREUR est affichée.

Syntaxe

```
SLOAD "nom"
```

Le nom est sensible à la casse.
8 caractères maximum.

Exemple

```
SLOAD "data"
```

Mots clés associés

CLOAD, CLOAD ?, SSAVE

SOUND

Objet

Une sortie de son est produite directement en envoyant les commandes au générateur de son. La sortie du son désiré peut être faite en désignant l'information (DATA) qui est dans le registre et les registres de 0 à 13 listés ci-dessous.

Syntaxe

```
SOUND registre,data
```

Registre :

Data :

Exemple

Mots clés associés

PLAY

SPC

Objet

Écrit le nombre spécifié d'espace.

Syntaxe

SPC (x)

x : nombre d'espace.

Exemple

```
10 PRINT"GAME";
20 PRINT SPC(3);
30 PRINT"OVER"
```

Place 3 espaces entre les 2 mots.

Noter le point-virgule en fin des lignes 10 et 20 pour continuer l'affichage sur la même ligne.

Les 3 lignes peuvent se fusionner.

```
10 PRINT"GAME";SPC(3);"OVER"
```

Mots clés associés

LOCATE, LPRINT, PRINT, PRINT#, TAB

SQR

Objet

Renvoie la racine carrée de x.

Syntaxe

SQR (x)

x doit être zéro ou positif

Exemple

```
PRINT SQR(10)
```

Résultat : 3.16227766

Mots clés associés

[DEF FN](#)

SSAVE

Objet

L'information affichée sur l'écran est sauvegardée sur la cassette du magnétophone.

SSAVE peut servir de commande ou d'instruction.

Avant l'exécution d'un SAVE, s'assurer que le magnétophone est correctement branché et que les touches RECORD et PLAY sont bien enfoncées.

Syntaxe

```
SSAVE "fichier"
```

Fichier : nom du fichier qui contiendra les informations de l'écran. Le nom est sensible à la casse.

Exemple

```
SSAVE "data"
```

Mots clés associés

CLOAD, CSAVE, SLOAD

STEP

Objet

Syntaxe

STEP

Exemple

```
10 FOR I=1 TO 10 STEP 2  
20 NEXT I
```

Mots clés associés

FOR, NEXT

STICK

Objet

Permet de tester les manettes et le clavier (touches de direction)
Nécessite d'avoir le synthétiseur pour les joysticks.

Syntaxe

STICK(x)

x=0 : Touches du clavier

x=1 : Joystick 1

x=2 : Joystick 2.

Exemple

```
10 IF STICK(1)=1 then print "nord"
20 IF STICK(1)=2 then print "nord-est"
30 IF STICK(1)=3 then print "est"
40 IF STICK(1)=4 then print "sud-est"
50 IF STICK(1)=5 then print "sud"
60 IF STICK(1)=6 then print "sud-ouest"
70 IF STICK(1)=7 then print "ouest"
80 IF STICK(1)=8 then print "nord-ouest"
90 IF STICK(1)=0 then print "manette inactive"
```

Mots clés associés

STRIG

STOP

Objet

Arrête le programme en exécution.

Le STOP peut être mis à n'importe quel endroit du programme pour arrêter son exécution. Toutefois, contrairement à l'instruction END, lorsque STOP est exécuté, le message suivant apparaît :

BREAK in nnnnn

Nnnnn étant le numéro de ligne où se situe le STOP.

Syntaxe

STOP

Exemple

9990 **STOP**

Mots clés associés

END, CONT

STRIG

Objet

Test du bouton de Joystick ou barre espace.

Syntaxe

STRIG (x)

x=0 : Barre espace.

x=1 : Bouton Joystick 1.

x=2 : Bouton Joystick 2.

Renvoie 1 si l'appuie est détecté, sinon 0.

Exemple

```
10 IF STRIG(0)=0 THEN GOTO 10
```

Attend l'appuie sur la barre d'espace.

Mots clés associés

STICK

STR\$

Objet

Transforme un nombre en chaîne de caractères.

Syntaxe

STR(x)

Exemple

```
10 X=10
20 PRINT STR(X)
```

Mots clés associés

[ASC](#), [CHR\\$](#), [DEF FN](#), [LEFT\\$](#), [MID\\$](#), [RIGHT\\$](#), [VAL](#)

TAB

Objet

Met des espaces jusqu'à la colonne spécifiée.

Syntaxe

TAB (x)

x : numéro de colonne.

Exemple

```
10 PRINT"XXX";TAB(10);"XXX"  
20 PRINT"PHC-25";TAB(10);"PROGRAMME"
```

Affichera la 2^{ème} partie du PRINT à partir de la colonne 10.

Mots clés associés

LPRINT, PRINT, PRINT#, [SPC](#)

TAN

Objet

Renvoie en simple précision la tangente de l'angle, exprimée en radians

Syntaxe

TAN (x)

x : angle dont on veut la tangente.

Exemple

```
PRINT TAN(1.34)
```

Renvoie : 4.67344123

Mots clés associés

COS, [DEF FN](#), SIN

TIME

Objet

Compteur de temps qui ajoute 1 chaque 1/360 secondes.
Le compteur à une précision faible.

Syntaxe

TIME

Exemple

```
10 A=TIME  
20 B=TIME-A  
30 PRINT INT(B/360)  
40 GOTO 20
```

Mots clés associés

RND

THEN

Objet

Suivant les résultats d'un test, l'on prendra une décision ou l'on choisira une suite pour l'exécution du programme.

Si le résultat du test est vrai, c'est-à-dire différent de zéro, alors l'instruction THEN suivra jusqu'à ELSE ou la fin de la ligne et sera exécutée.

Si l'expression est égale à zéro (valeur fausse), seules les instructions suivant ELSE, si elles existent, seront exécutées.

Syntaxe

```
IF <expression> THEN <instruction> ELSE <instruction>
```

Exemple

```
5 A=-5: B=-10
10 IF A<>0 THEN B=10 ELSE B=0: GOTO 100
30 PRINT A,B
40 END
100 PRINT A,B
110 A=A+1
120 GOTO 10
```

Mots clés associés

ELSE, IF

USR

Objet

Exécute un programme fait par l'utilisateur en langage machine (Z80).

Syntaxe

USR (x)

Exemple

NDR : Aucune information disponible.

Mots clés associés

DATA, EXEC, PEEK, POKE, READ, RESTORE

VAL

Objet

Renvoie la valeur de la chaîne de caractères fournie en argument.

Si le premier caractère de la chaîne n'est pas +, -, &, ou un chiffre, la valeur renvoyée est 0.

En notation hexadécimale [0-9],[A-F], les autres caractères sont ignorés.

Syntaxe

VAL (x\$)

x\$: Chaîne à convertir

Exemple

```
10 FOR I=1 TO 3
20 READ A$
30 PRINT VAL("&H"+A$);
40 NEXT
50 DATA C3,40,FF
```

Lit les valeurs dans la ligne DATA et les converti depuis le format hexadecimale vers le format decimal.

Mots clés associés

[ASC](#), [CHR\\$](#), [LEFT\\$](#), [MID\\$](#), [RIGHT\\$](#), [STR\\$](#)

Mémoire du PHC-25

Plages mémoires

Zone	Adresse	
Travail du Basic	&hFFFF	
	&hF800	
Page vidéo 2	&hF7FF	
	&hE000	
Travail du programme	&hDFFF	
	&hC000	
Espace libre	&hBFFF	
	&h8000	
Espace libre	&h7FFF	
	&h7800	
Page vidéo 1	&h77FF	
	&h69FF	Screen 1,2 Plage 2
	&h6800	
	&h61FF	Screen 1,2 Plage 1
	&h6000	
Interpréteur BASIC	&h5FFF	
	&h0000	

Page Vidéo

Il y a 2 pages vidéo qui correspondent au choix fait lors du démarrage de la machine.

La page 1 est toujours utilisée.

La page 2 lorsqu'elle n'est pas utilisée et utilisée comme « Travail du basic ».

Chaque page vidéo contient 6Ko.

Pour plus de détail sur ces pages consulter l'instruction SCREEN et le chapitre consacré à l'écran.

Travail d'analyse plus poussé à faire.

NDR : Décodage de la RAM et de la ROM.

L'écran du SANYO PCH-25

Les 2 choix au boot

Lors du démarrage de la machine, apparaît :

SCREEN ?

La réponse est soit 1 soit 2.

À ne pas confondre avec l'instruction SCREEN.

Réponses :

1 – Un seul écran de travail

2 – 2 écrans de travail

La bascule entre les 2 écrans se fera avec CTRL+Q.

Les résolutions d'écran

Le tableau ci-dessous donne les possibilités du PHC-25.

a	1	2	3	4
Texte	16L 32C	16L 32C	16L 16C	16L 32C
Graphique	32x16	64x48	128x192	256x192

Les coordonnées texte commencent en haut à gauche en 0,0.

Les coordonnées graphiques commencent en haut à gauche en 0,0.

La première position est l'axe des abscisses.

La deuxième position est l'axe des ordonnées.

Exemple :

```
10 LINE(0,0)-(255,0)
```

Trace une ligne en haut de l'écran.

Selon le SCREEN choisi, la dimension de la ligne changera.

Pour les instructions graphiques, il faut toujours se référer à la dimension maximum, soit 256x192 pixels.

Caractères non affichables par CHR\$

Les caractères spéciaux de &h10 à &h1C (voir table page 75 de la documentation) ne sont pas affichable par PRINT CHR\$(n).

Il faut donc passer par la solution du POKE.

Le screen 1

Le screen 1 est de 32 colonnes sur 16 lignes.

Il s'agit d'un mode texte uniquement. Son adresse mémoire va de &h6000 à &h61FF.

Soit 512 octets, ce qui correspond bien à 32x16 caractères.

La numérotation par du coin en haut à gauche de l'écran, de 0 à 31 et 0 à 15.

Plages d'adresses vidéo

Table d'adressage (1.1) simplifiée (Data) :

Ligne	Colonne 0	Colonne 31
0	&h6000	&h601F
1	&h6020	&h603F
2	&h6040	&h605F
3	&h6060	&h607F
4	&h6080	&h609F
5	&h60A0	&h60BF
6	&h60C0	&h60DF
7	&h60E0	&h60FF
8	&h6100	&h611F
9	&h6120	&h613F
10	&h6140	&h615F
11	&h6160	&h617F
12	&h6180	&h619F
13	&h61A0	&h61BF
14	&h61C0	&h61DF
15	&h61E0	&h61FF

Il est possible de simuler le LOCATE x,y directement sur l'écran avec un POKE directement dans l'adresse avec la valeur du caractère souhaité.

La valeur de l'octet mis à l'écran correspond au générateur de caractère (voir page 75 de la documentation) ou le chapitre « Jeu de caractères ».

Chaque partie de l'écran est donc représentée par un octet.

En principe cet octet est de type ASCII, mais réduit du fait de la simplification du générateur de caractère (version EU).

Les couleurs sont dépendantes de la deuxième plage d'adresse (&h6800 - &h69FF).

L'octet de cette plage va influencer la couleur et si on passe sur du texte ou du pseudo graphique.

Table d'adressage (1.2) simplifiée (Attributs) :

Ligne	Colonne 0	Colonne 31
0	&h6800	&h681F
1	&h6820	&h683F
2	&h6840	&h685F
3	&h6860	&h687F
4	&h6880	&h689F
5	&h68A0	&h68BF
6	&h68C0	&h68DF
7	&h68E0	&h68FF
8	&h6900	&h691F
9	&h6920	&h693F
10	&h6940	&h695F
11	&h6960	&h697F
12	&h6980	&h699F
13	&h69A0	&h69BF
14	&h69C0	&h69DF
15	&h69E0	&h69FF

Cependant, les éléments pseudos graphiques seront toujours sur fond noir. Cela nous permet d'avoir des couleurs supplémentaires sur le screen 1.

En fonction des bits des 2 octets, l'affichage va donc varier.

Plage d'adressage 1 : Type de caractère (texte ou pseudo graphique) dit Data.

Plage d'adressage 2 : Couleur et Texte ou pseudo graphique, dit Attributs.

Table de couleur des textes ou graphique (G).

Adressage 2 (&h6800)		Adressage 1 (&h6000)			Couleur
&x----	&x0000	&h00	&x----	&x----	0
	&x0001	&h01			1
	&x0010	&h02			G
	&x0011	&h03			G
	&x0100	&h04			2
	&x0101	&h05			3
	&x0110	&h06			G
	&x0111	&h07			G
	&x1000	&h08			0
	&x1001	&h09			1
	&x1010	&h0A			G
	&x1011	&h0B			G
	&x1100	&h0C			2
	&x1101	&h0D			3
	&x1110	&h0E			G
	&x1111	&h0F			G

On constate une répétition

Nous sommes sur un adressage 11 bits pour le mode écran 1.

&x 1??? ?c0i nnnn nnnn

Concernant l'affichage de texte :

- Les bit 0 à 8 sont la représentation des caractères ASCII.
- Le bit numéro 10 toujours à 0.
- Les bits 9 et 11, vont faire varier les 4 couleurs disponibles de la palette (cf. COLOR).
 - Le bit 9 (i) provoque l'inversion vidéo.
- Le bit 16 à 1.
- Les autres bits (?) ne semblent pas avoir d'influences.

Mode 1 et pseudo graphique

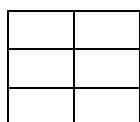
Le mode impossible, mais pas français...

Il y a 512 cases disponibles qui ont la dimension d'un caractère.

Ces cases ne peuvent avoir que 2 couleurs dont obligatoirement le noir (le fond).

L'autre couleur étant choisie dans la palette (8 possibles, voir COLOR).

Les cases ont une dimension de 2x3.



Subdivisé à la résolution :

O	O	O	O	O	O	O
O	O	O	O	O	O	O
O	O	O	O	O	O	O
O	O	O	O	O	O	O
O	O	O	O	O	O	O
O	O	O	O	O	O	O
O	O	O	O	O	O	O
O	O	O	O	O	O	O
O	O	O	O	O	O	O
O	O	O	O	O	O	O
O	O	O	O	O	O	O
O	O	O	O	O	O	O

Chaque « O » est un pixel de la résolution de 256x192.

Si la figure ci-dessus représente le coin haut, gauche :

- Elle couvre les points de (0,0) à (7,11) soit un bloc 96 pixels.
- L'adresse de cette case mémoire vidéo est &h6000.

Un POKE sur &h6000 va donc allumer 0 à 6 des portions avec une des 9 couleurs selon la palette (1 ou 2, voir [COLOR](#)).

Le codage pseudo graphique utilise l'adressage vu précédemment.

2 octets aux plages d'adresse (&h6000-&h61FF)-(&h6800-&h69FF).

Soit l'octet &xnnnnnnn. Dont nous trouvons les bits dans la matrice de résolution vue ci-dessus.

5	4
3	2
1	0

Ces 6 bits sont sur la plage (&h6000-&h61FF).

Valeurs des 6 bits :

- 0 : zone éteinte, couleur noir)
- 1 : zone allumé, couleur en fonction des autres bits

Il y a 9 couleurs disponibles, cela implique 3 bits pour le codage. Vue précédemment, l'encodage est sur 2 octets. Soit 16 bits. Après analyse et décryptage ci-dessus, le décodage donne :

- n : bit graphique
- x : bit sans influence
- c : bit pour la couleur

&h1xxx 0c1x ccnn nnnn

Il y a donc 64 pseudos graphiques bicolor.

0 ; 4 ; 8 ; C

2 ; 6

TO DO :

Plage d'adressage 1 et 2 en mode 2 écran.

- &hE000 - &hE1FF (probable 2.1)
- &hE800 - &hE9FF (probable 2.2)

Instructions graphiques

Il est possible de tracer des lignes, des rectangles, des rectangles pleins, etc. avec les fonctions BASIC. Il faut cependant considérer que la résolution est de 32x16 pixel.

Ligne et point sont des gros « pâtés » de la taille d'un caractère.

PSET et PRESET provoquent des effets de bord sur l'écran qui doivent être analysés.

Ces instructions modifient l'état des octets des 2 plages d'adresse du mode.

Il est recommandé de ne pas les utiliser.

Exemples

Exemple 1 :

```
1000 REM Poke direct sur le screen 1
1010 SCREEN 1,1,1: CLS
1020 A=0
1030 FOR I=&h10 to &h85
1040 POKE &h6000+A,I
1050 A=A+1
1060 NEXT I
1070 GOSUB 9980
1080 END
9980 K$="" :K$=INKEY$:IF K$="" THEN 9980
9990 RETURN
```

Ce programme affiche les caractères de **&h10** à **&h85** sur un screen 1.

Ce principe permet d'aller assez vite pour afficher à l'écran car on écrit directement dans la mémoire vidéo du PHC-25.

Exemple 2 :

Pour faire un LOCATE 10,10 (colonne 10, ligne 10).

L'adresse de la ligne 10 est **&h6140**.

La colonne 10 est un incrément de **&hA**.

La localisation est donc **&h614A**.

```
POKE &h614A,&h2A
POKE &h6140+&hA,&h2A
POKE &h6140+10,&h2A
```

Le résultat de ces 3 lignes est identique.

Affichera « * » en colonne 10 ligne 10.

L'instruction CONSOLE ne bloquera pas le POKE.

Le Screen 2

Description

Le Screen 2 est un mixe entre mode texte et mode graphique. La résolution texte est de 32 colonnes et 16 lignes. La résolution graphique est de 64x48 pixels.

Un graphique ne peut pas s'afficher correctement sur du texte déjà présent à l'écran.

Le fond pour la partie graphique doit toujours être BLACK (valeur 0).

Il est conseillé d'utiliser [CONSOLE](#) pour définir les lignes sur lesquelles mettre du texte ou du graphique.

Il y a 2 palettes de couleurs disponibles (voir [COLOR](#)). Un changement de palette affecte l'écran entier.

Concernant le texte, il est possible de faire des **PRINT CHR\$(n)**, sauf des caractères spéciaux de **&h10** à **&h1C**.

La mémoire vidéo est la même qu'avec le SCREEN 1.

- **&h6000** à **&h61FF**. Plage d'adressage 1 (Data).
- **&h6800** à **&h69FF**. Plage d'adressage 2 (Attributs).

Ce mode est recommandé car il apporte plus de possibilité que le mode 1.

Cependant, faire attention au texte qui n'a pas de fond noir.

Graphique en mode 2

L'instruction SCREEN 2, met l'écran en mode 2 et effectue un **CLS** sur fond noir.

Le fonctionnement est identique à ce que nous avons vu pour le mode 1. Le fond étant noir.

Subdivisé à la résolution :

O	O	O	O	O	O	O
O	O	O	O	O	O	O
O	O	O	O	O	O	O
O	O	O	O	O	O	O
O	O	O	O	O	O	O
O	O	O	O	O	O	O
O	O	O	O	O	O	O
O	O	O	O	O	O	O
O	O	O	O	O	O	O
O	O	O	O	O	O	O
O	O	O	O	O	O	O
O	O	O	O	O	O	O
O	O	O	O	O	O	O
O	O	O	O	O	O	O
O	O	O	O	O	O	O
O	O	O	O	O	O	O

Chaque « O » est un pixel de la résolution de 256x192.
Si la figure ci-dessus représente le coin haut, gauche :
Elle couvre les points de (0,0) à (7,11) soit un bloc 96 pixels.

Un **PSET** mis sur 0,0 allume tous les pixels de (0,0) à (3,3) et affecte la couleur sur la zone (6 blocs).

L'adresse de cette case mémoire vidéo est **&h6000**.

Un **POKE** sur **&h6000** va donc allumer 0 à 6 des portions avec une des 9 couleurs selon la palette (1 ou 2, voir **COLOR**) et uniquement sur fond noir.

Quand 2 lignes graphiques se croisent, à l'intersection, la case prend la couleur de la dernière ligne qui s'y affiche Pour les pixels déjà allumés (ils font 4x4).

Positionner un pixel (PSET) implique la mise à la même couleur des pixels déjà présent (0 étant non présent).

POKE mémoire vidéo

Comme en SCREEN 1 (voir description du screen 1).

Rappel :

Octet gauche : **&h6800** à **&h69FF**.

Octet droit : **&h6000** à **&h61FF**

&x 1--- -c0i nnnn nnnn

- Les bit 0 à 8 sont la représentation des caractères ASCII.
- Le bit numéro 10 toujours à 0 pour ASCII.
- Les bits 9 et 11, vont faire varier les 4 couleurs disponibles de la palette (cf. **COLOR**).

&h 1xxx xc1x ccnn nnnn

- Bits 0 à 6, schéma graphique
- x : bit sans influence
- c : bit pour la couleur
- bit 16 : toujours à 1.
- Bit 10 : Toujours à 1 en graphique

Le bit no 10 de la plage d'adresse étant à 1, il faut le mettre à 0 pour afficher les caractères de **&h10** à **&h1C**.

Il peut être intéressant pour un affichage texte de positionner le bit 10 à 1, puis par programme le positionner à 0 pour créer une animation type déchiffrement du texte.

Le Screen 3

Description

Le SCREEN 3 offre une résolution graphique de 128x192 pixels.

En mode texte, la résolution est de 16x16 caractères.

Il y a 4 couleurs disponibles sur un choix de 2 palettes (voir [COLOR](#)).

En fin de programme, le PHC-25 retourne automatiquement en mode 1.

Ce qui implique que tout l'écran est perdu.

D'où le mode 2 écran du PHC-25, afin de ne pas perdre les tracés au détriment de la mémoire disponible. Changement par les touches « Control+Q ».

Il n'est pas possible d'utiliser l'instruction [INPUT](#), cela provoque un retour en mode screen 1 (mode texte), et implique que tous les graphiques sont perdus.

Plage d'adresse

De &h6000 à &h77FF pour l'écran 1.

De &hE000 à &hF700 pour l'écran 2.

Particularité des pixels

Les pixels sont plats.

Les coordonnées commencent en haut à gauche au point (0,0).

TO DO : encodage d'un pixel.

Le Screen 4

Description

Le SCREEN 4 offre une résolution graphique de 256x192 pixels.

En mode texte, la résolution est de 32x16 caractères.

Il y a 2 couleurs disponibles (voir [COLOR](#)).

Plage d'adresse

De &h6000 à &h77FF pour l'écran 1.

De &hE000 à &hF700 pour l'écran 2.

Particularité des pixels

Les pixels sont carrés.

Les coordonnées commencent en haut à gauche au point (0,0).

TO DO : encodage d'un pixel.

Jeu de caractères

Les caractères

Les caractères sur le SAYNO PHC-25 sont dans une matrice de 8x12 pixels.

Adresse mémoire d'encodage des caractères.

&h ????.

Le générateur de caractères

De &h00 (00) à &h0F (15)

TO DO.

À &h10 (16)

Le caractère de PI.

NDR : il n'y a pas d'instruction PI.

Ce caractère n'est pas affichable via PRINT CHR\$(n). Cependant il est affichable via un [POKE](#) à l'adresse mémoire vidéo des mode 1 et 2.

Il est affichable via la combinaison de touche []+[].

De &h11 à &h1B

Les éléments pour tracer des tableaux en mode texte.

Hexa	11	12	13	14	15	16	17	18	19	1A	1B
Déc	17	18	19	20	21	22	23	24	25	26	27
Car.	⊥	⊤	⊣	⊢	⊕	⊖	—	⊸	⊴	⊶	⊷

Tous ces caractères ne sont pas affichables via PRINT CHR\$(n). En revanche ils sont affichables via un POKE à l'adresse mémoire vidéo des mode 1 et 2.

Ils sont affichable via la combinaison de touche []+[].

À &h1C (28)

Une croix bien centrée (X), différente du x (majuscule ou minuscule).

Ce caractère n'est pas affichable via PRINT CHR\$(28). Cependant il est affichable via un POKE à l'adresse mémoire vidéo des mode 1 et 2.

Il est affichable via la combinaison de touche []+[].

De &h20 (32) à &h7F

Caractères ASCII alphanumériques standard.

De &h80 (128) à &h83 (131).

Les caractères pique, cœur, trèfle et carreau.

Hexa	80	81	82	83
Déc	128	129	130	131
Car.	♠	♥	♣	♦

Ces caractères sont affichables avec PRINT CHR\$(n).

De &h84 à &h85

- Rond vide (○),
- Rond plein (●).

Ces caractères sont affichables avec PRINT CHR\$(n).

Toutes les autres valeurs afficheront un espace vide.

NDR : à revoir pour la ROM Japonaise.

Hack de la ROM ?

Il y a un nombre d'espace vide assez conséquent sur la ROM PAL. Ne pas oublier qu'à l'origine c'est un ordinateur Japonais et qu'il avait besoin d'un jeu de caractères large.

N'ayant pas vérifié la documentation japonaise, le sujet est en suspens.

Cependant, rien n'empêche de « trafiquer » la ROM PAL pour ajouter un nombre conséquent de caractères au PHC-25.

Mais c'est un autre chapitre du PHC-25.

Trucs et Astuces

Attendre l'appuie sur une touche

```
1000 K$="" : K$=INKEY$ : IF K$="" THEN 1000  
1010 RETURN
```

Un GOSUB 1000, fera un saut ligne 1000, il y aura alors attente jusqu'à ce que l'utilisateur appuie sur une touche.

K\$ prend la valeur de cette touche. K\$ peut être exploitée pour la suite du programme après l'instruction RETURN.

RANDOMIZE

Comme vu dans la description de la fonction RND. Donner une valeur négative réinitialise la séquence.

```
1000 I=RND (-TIME)  
1010 RETURN
```

En utilisant TIME, le seed sera en principe toujours différent.

Il est possible de lancer le sous-programme plusieurs fois durant l'exécution du programme principal.

Forcer 1 ou 2 écrans

Ceci peut s'avérer nécessaire dans le cas où le programme doit utiliser 2 écrans. Typiquement lors de tracé de courbe mathématique ceci afin de ne pas perdre le tracé.

Test si 1 ou 2 écrans

TO DO

Bascule 1 vers 2 et inversement

2 vers 1

```
10 poke &HFB58,247  
20 poke &HFB56,1  
30 clear 50,&HF800
```

1 vers 2

```
10 clear 50,&HE000  
20 poke &HFB58,223  
30 poke &HFB56,20
```

La différence et l'allocation ou pas de l'espace mémoire de la mémoire vidéo 2.

La fonction MOD

La fonction MOD est le modulo (le reste) d'une division mathématique.

Le PHC-25 n'a pas cette fonction.

Pour simuler cette fonction :

Soit la formule :

x = y MOD z

En BASIC PHC-25 écrire :

x = y - (INT(y/z)*z)

La fonction de division entière (\)

Cette fonction n'est pas disponible dans le PHC-25.

La fonction permet d'obtenir le reste de la division de Y/Z.

Soit la formule :

x = y \ z.

En BASIC PHC-25 écrire :

```
x = INT(y/z)
```

Majuscules et Minuscules

Le PHC-25 ne dispose pas des fonctions classiques UPPER\$ et LOWER\$ que l'on trouve dans d'autres BASIC.

Les sous-programmes suivants pallient ce manque.

```
1000 REM This routine converts A$ to uppercase
1010 REM Loop to process each character
1020 REM ASCII check and conversion
1030 FOR I=1 TO LEN(A$)
1040   C$=MID$(A$,I,1)
1050   X=ASC(C$)
1060   IF X>=97 AND X<=122 THEN C$=CHR$(X-32)
1070   A$=LEFT$(A$,I-1)+C$+RIGHT$(A$,LEN(A$)-I)
1080 NEXT I
1090 RETURN
```

```
2000 REM This routine converts A$ to lowercase
2010 REM Loop to process each character
2020 REM ASCII check and conversion
2030 FOR I=1 TO LEN(A$)
2040   C$=MID$(A$,I,1)
2050   X=ASC(C$)
2060   IF X>=65 AND X<=90 THEN C$=CHR$(X+32)
2070   A$=LEFT$(A$,I-1)+C$+RIGHT$(A$,LEN(A$)-I)
2080 NEXT I
2090 RETURN
```

Placer la chaîne à convertir dans A\$, appeler la sous routine. A\$ sera changé.

Utilise une chaîne intermédiaire C\$.

Utilise deux variables I et X.

Ces trois variables sont donc changées durant le processus, donc attention lors de l'adaptation.

Si le caractère n'est pas alphabétique le sous-programme le reporte tel quel.

Algorithmes à tester et améliorer. Garbage sur les intermédiaires pour éviter l'overflow mémoire.
Possible changement dans une version futur de ce document.

Conversion LOCATE et adresse mémoire vidéo

Algorithme pour transformer un LOCATE en Adresse vidéo et inversement.

Pour SCREEN 1 et 2.

X,Y vers mémoire

Pour obtenir l'adresse à partir de (x, y), on prend la base **&H6000**, puis on ajoute ($y \times 32$) + x (chaque ligne contient 32 caractères).

Par calcul :

```
1000 REM Convertir (X,Y) en adresse mémoire
1010 REM Entrée : X,Y
1020 REM Sortie : AD
1030 AD=&H6000+(Y*32)+X
1040 RETURN
```

Par fonction DEF FN :

```
1000 REM Convertir (X,Y) en adresse mémoire
1010 DEF FN AD(X,Y)=&H6000+(Y*32)+X
```

Problème avec le DEF FN non fonctionnel sur le PHC-25 (NDR : contrairement à ce que dit la notice, sauf si problème sur émulateur).

Mémoire vers X,Y

Pour retrouver x à partir d'une adresse, on fait MOD 32, qui nous donne la position horizontale. Pour retrouver y, on divise (\ division entière) par 32 pour obtenir la ligne correspondante.

Par calcul :

```
2000 REM Convertir adresse mémoire en (X,Y)
2010 REM Entrée : AD
2020 REM Sortie : X,Y
2030 X = (AD - &H6000) MOD 32
2040 Y = (AD - &H6000) \ 32
2050 RETURN
```

Par fonction DEF FN :

```
2000 REM Convertir adresse mémoire en (X,Y)
2010 DEF FN X(AD)=(AD-&H6000) MOD 32
2020 DEF FN Y(AD)=(AD-&H6000) \ 32
```

Mais notre BASIC PHC-25 n'a ni MOD ni la division entière () .

Nous l'avons vue précédemment. Il faut donc s'adapter.

Par calcul :

```

2000 REM Memory Address to (X,Y)
2010 REM IN : AD
2020 REM OUT : X,Y
2030 X = (AD-&H6000)-(INT((AD-&H6000)/32)*32)
2040 Y = INT((AD-&H6000)/32)
2050 RETURN

```

Par DEF FN :

```

2000 REM Memory Address to (X,Y)
2010 REM AD: Address to convert
2020 REM X and Y, equal LOCATE X,Y
2030 DEF FN X(AD)=(AD-&H6000)-(INT((AD-&H6000)/32)*32)
2040 DEF FN Y(AD)=INT((AD-&H6000)/32)

```

Tests des adresses adjacentes

```

AU=AE-32 : REM UP
AD=AE+32 : REM DOWN
AL=AE-1  : REM LEFT
AR=AE+1  : REM RIGHT

```

Tester les valeurs

Limites

Si AE<&h6000 alors hors écran

Si AE>&h61FFF alors hors écran

Tests des bords, gauche et droite à faire. En effet un +1 sur le bord droit fait passer à la ligne suivante.

Conversion DEG et RAD

Le PHC-25 ne connaît que les Radians.

La variable PI n'existe pas dans le BASIC du PHC-25, il est recommandé de la créer s'il y en a besoin dans votre programme.

Voici les conversions.

RAD vers DEG

```
10 PI=3.141592653: REM Define PI  
1000 D = R * (180 / PI)
```

Formule que l'on peut placer dans un DEF FN.

```
10 PI=3.141592653: REM Define PI  
1000 DEF FN D(R)=R*(180/PI)
```

Utilisation.

```
2000 X=FN D(PI/2)
```

Converti PI/2 en radian, X vaudra 90°.

DEG vers RAD

```
10 PI=3.141592653: REM Define PI  
1000 R = D * (PI / 180)
```

Formule que l'on peut placer dans un DEF FN.

```
10 PI=3.141592653: REM Define PI  
1000 DEF FN R(D)=D*(PI/180)
```

Utilisation.

```
2000 X=COS(FN R(45))
```

Calcul le cosinus de 45° et le place dans X.

Formules Graphique

Tracé de cercle

Cercle par formule.

TO DO

Cercle rapide.

TO DO

Tracé d'ellipse

TO DO

INP et OUT

La documentation est vierge sur ce domaine. Cependant voici quelques informations.

Elle sont issues du GitLab :

https://gitlab.com/mokona/phc25_tools/-/blob/main/documentation/phc25-tech-information.md

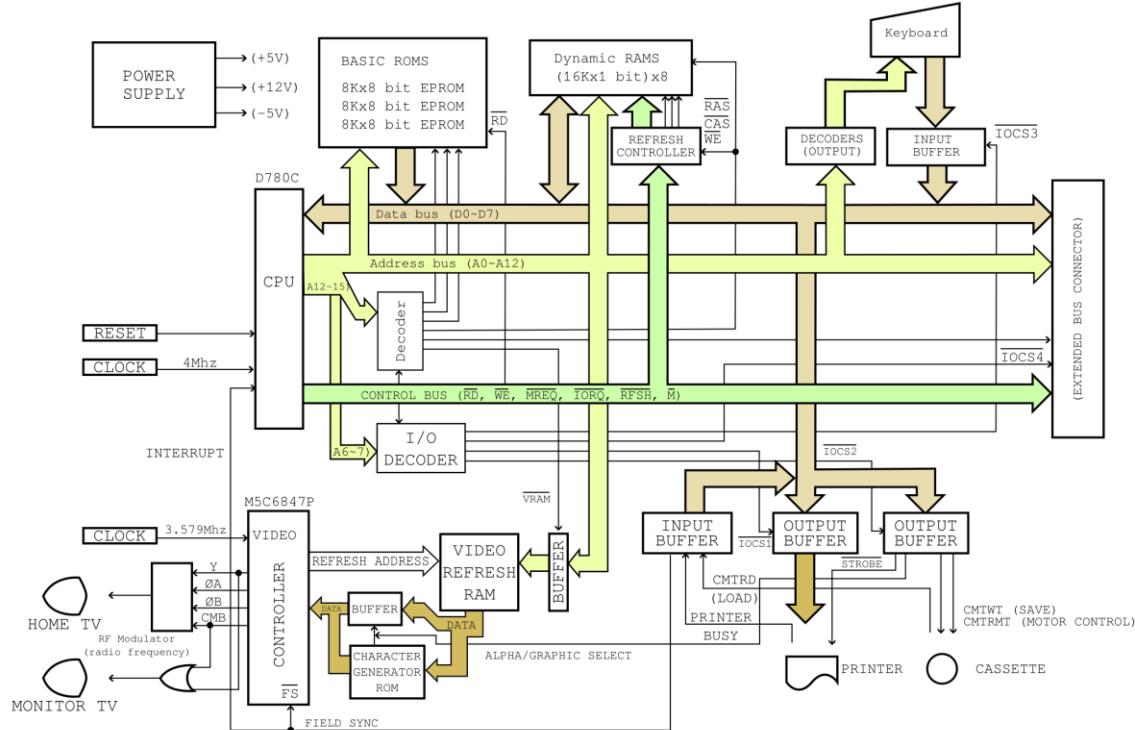
Port I/O

Port	Description
&h00	Port Centronics (imprimante, écriture)
&h40	Lecture/écriture
&h80-&h88	Lecture Clavier
&hC0	AY8910 (son), écriture
&hC1	AY8910 (son), lecture/écriture

Dossier technique

Block Diagram

SANYO PHC-25 BLOCK DIAGRAM



Redone from documentation
Version: 1.0

Source :

https://gitlab.com/mokona/phc25_tools/-/blob/main/documentation/phc25-tech-information.md

Les connecteurs



De gauche à droite :

Alimentation, Extension, Imprimante, Vidéo, Inutilisé, Peritel, Magnétophone.

Alimentation

La prise est une fiche IEC 60320 C8 mâle.

Câble électrique broche Type C (Europlug) (2 broches) mâle pour la version EU d'un côté et une IEC 60320 C8 (2 broches) femelle de l'autre côté.

Extension

Fiche IDC 34 broches

Pin	Signal	Pin	Signal
1	D0	2	Masse (GND)
3	D2	4	
5		6	
7	A12	8	
9	A10	10	A11
11	A6	12	A9
13	A8	14	A7
15	A4	16	A5
17	A2	18	A3
19	A0	20	A1
21	IOCS4	22	
23	D6	24	
25	D4	26	
27	IORQ	28	
29	RESET	30	
31	Vcc	32	Vcc
33	Masse (GND)	34	Masse (GND)

Imprimante

Fiche Centronics (14 broches)

NDR : Ces fiches ne sont pas normalisée IEC.

Prise Vidéo

Inutilisé bleu

On ne sait pas pourquoi c'est là.

Prise vidéo Péritel

Il s'agit d'une fiche DIN 8 broches

Pin	Signal	Couleur
1	Bleu	
2		
3	Masse	
4	Rouge	
5		
6	Vert	
7	+12V	
8		

Magnétophone

Il s'agit d'une fiche DIN 8 broches

Pin	Signal	Couleur
1	Masse	
2	Masse	
3	Masse	
4		Rouge
5		Blanc
6		Noir
7		
8	Masse	

Fin du livre.

4^{ème} de couverture à faire.