conquérant

brouillon

réglure 2 couleurs

96 PAGES

Nº 2160

Sinon ERROR
Il y 1359 (ou 1539) ?
ou une not opisrel ?
1539 ?

POKE

Commande à utiliser avec précaution !
- chapitre 3 page 58 -

DÉCRÉMENTER = Diminuer (val -)
INCRÉMENTER Augmenter (val +)

CURSEUR comment le placer — (locate) + CHR$

Comment replacer une ligne en face de son chiffre —

ex { 40 -- variable
{ 50 -- "

voir RENUM. chap 3 p 66 —

voir les commandes en CPM (effacement etc).

~~Relecture du texte~~ en LOGO ed ~~et correction~~ —

voir pour la lecture détaillée du CAT —

chap. 6 p 12 se — comment trouver le (—)?

pour revenir de EDIT à normal taper (CONTROL) ESC —

CLEAN efface l'ÉCRAN — sauf TORTUE

— La fonction ~~ER~~ EDALL recharge l'écran (~~copie CA~~ ?)

pour faire réapparaître le dessin après CONTROL ESC —

Repasser ed et ~~COPY~~ RETURN — passe en fonction EDIT —

Taper ESC — passer en ? écran libre (sans EDIT)

en tapant (CONTROL?) ESC — pour stopper

ou COPY et le graphique se fait

puis CARRE ou TRIANGLE et

ed RETURN l'écran se met en EDIT pour correction

sans afficher le texte, mais ~~les~~ le texte des dessins (carré / triangle etc)

EDALL part du texte origine pour se présenter en

EDIT ~~pour~~ avec le texte pour correction

s'il disparait / CONTROL f et ↑ (curseurs)

taper CARRE / TRIANGLE

et le dessin s'affiche — (se forme)

X = 1 à
1 20 Y = 1 à

05.20

Dr LOGO.

Tapant ed [RETURN] l'écran se met en EDIT sous affichage de Texte + (Ayant tapé ESC)
pour faire venir ce Texte en EDIT pour correction - taper EDALL - / Il passe en EDIT avec le Texte - Faire correction avec COPY
Pour l'affichage des formes définies (Triangle / carré défini réel) taper - lorsque l'écran est en EDIT la fonction COPY + Taper la forme écrite à l'écran et le graphisme répond - (ex: Triangle)
(Attention ERALL efface les variables)

~

en ASCII Mode 1 - 40 Largeur / 25 - hauteur
CURSEUR Mode 0 = 20 " / 25
Mode 2 = 80 " / 25

Placement - 10 - LOCATE (n° de colonne entre - et 9 (n° de ligne entre et
20 PRINT.

Mode 1 Mode 0 Mode 2
Curseur = 10 LOCATE / 1 à 40 ou / à 20 / à 80
Caractère colonne —

0 à 25
Soit en Mode 1 - 10 LOCATE n° Colon / n° Ligne
20 Print CHR$ (x) -
RUN -

(x) à 255 -)

EMS ?

Colonne 20 à 40 ?
Milieu Exp ?

1 à 40 en →

26
26
52

26
26
52

260
260
520

Pour affichage du Cat. actuelle Page 1 –

TYPE – KEYS – CCP – ? AMSDOS.COM /

BANKMAN.BAS / – BANKMAN.BIN / CLOCPM3.EMS / DATE.COM /
DEVICE COM / DIR.COM / DISCKIT3.COM / ED.COM / ERASE COM – /
GET.COM / KEY.CCP / KEY.WP / LANGUAGE.COM / PALETTE.COM /
PIP.COM / PROFILE.ENG / PUT.COM / RENAME.COM /

LOGO – Limites à partir du centre de la Tortue

↑ Fd – fd 190 –
→ Fd 310 – rt 90 – coin haut
↗ Fd ~~268~~ ~ ~~220~~ 265 –
rt 45

Cadre

hauteur – Fd 300 ~ 1200 –
Largeur – 620 ~ 16 x 14 –

diagonale – ?

En ASCII 132 localisation du Curseur (LOCATE)
et des pixels. (PLOT)

LARGEUR du Tableau

PIXEL LOGO

DOT. Cercle – en LOGO ?

PLOT

DOT E50 10

0101 0101 –
$2^6 0 2^4 0 2^2 0 1$ = 85 déci –
85 déc = 55 en Hex. $(16 \times 5) + 5$

33 à 47 / 56 à 64 / 93 ~~98~~ 91 à 96 /
ponctuation et signes Maths
chiffre 43 à 57
Alphabet MAJUSCULE 65 à 90
Minuscules 97 à 122
alphabet accentué

Revoir 160

Positionnement
d'un caractère sur l'écran —
en ASCII

10 - LOCATE x, y (1)*

20 - PRINT. CHR$ (32 à 255) suivant le caractère désiré

(1) x = colonne - de 1 à 40 en Mode 1.)
x = " de 1 à 20 en Mode 0)
x = " de 1 à 80 en Mode 2 -
y de 0 à 25 tous les modes -

A 10
B 11
C 12
D 13
E 14
F 15

~ LOGO Fd ou
Longueur / largeur

Pour les PIXELS -

PLOT x, y (RETURN) rt angle / supplémentaire
x = 1 à 639 - gauche à droite -
y = 399 à 1 de haut en bas

Tracer un trait d'un point (pixel à un autre)
DRAW - x, y soit 320, (centre) 1 (bas)
puis draw 1. à 390 et - 160

MOVE place le pixel mais non visible -
dessin plus lent. +
ORIGIN idem + rapide

Pr LOGO (rt = nbre de tour)

Cercle paramètres (Fd rayon ou ∅)
To cercle mais les
repeat 360 [Fd. rt.] paramètres
ou 36 (plus rapide doivent
mais x 10 les se tenir
un point décimal après paramètres
paramètre ex: Fd 1.9.
rectangle 4 [Fd - rt - Fd] repeat - de 1,1 ex
le cercle "s'ovalise" avec ~~st sft~~ setscrunch / et sf

PRGR DEM.

n° 28 - HORLOGE / CONVERSION Mètre / MHz ?
n° 30 Distance / AZ _ / SERRE LISTE ?
n° 31. Fréquency reception guide ?
n° 32 Distance / AZ ?

LOGO. Etoile.
to Etoile _ (5 branches)
repeat. 5 [Fd. (longeur / rt 144 retour angle
Fd (longueur / Lt 72 retour d'angle _

rappel du tapis sans Edit.
Text "on / po "" "cercle / carré / Triangle etc _
bien faire gaffe aux guillemets (efface

Pour placer la Tortue sur l'écran +
inplacer [pu] avoir paramètre / (fd . rt etc) puis pour revenir au trait visible _ chefer [pd] ou setpos [x y] ?

Pour Couvrir la Fenêtre (WINDOW) (0 elle disparait) _
- setbg (0 à 3) (0 = bleu par défaut 1 = blanc _ ?
- cs 2 = bleu pale / 3 = rouge
Pour "refermer" setbg 0 ? (ou sur la même logne
cs _

ht la flèche disparait. st réapparait

cercle + sf ~~setscrunch~~ + sf ? Etoile =
= paramètre 9 [fd 120 rt 160]
une série de traits - à revoir

- make "x 5 donne une valeur à la variable
- cs - vide l'écran
- st - rend visible la Tortue
- pd - le stylo reprend son trait.

repeat 30 [fd :x rt 90 make "x :x + 5] (LAB)
dicte le graphique de +5 en +5

fd 20 rt 90 Trait / Angle.
fd 10 Trait.
pu - lève le stylo (ne trace pas)
home ramène la Tortue au centre
bk 2 arrière 2 -
home ramène Tortue au centre
bk 2 arrière 2
pd - le stylo reprend son trait
setpc 2 donne le couleur au stylo - (Fill sur la même ligne ?)
Fill. (rejeter sur le p) ?? -

FILL?

en ASCII

L'on place le curseur au ~~centre~~ centre de 3E = 62
le fig + puis Fill - (n°)
- l'on place la Tortue au centre - dot.

/100/-

Nombres ~~HEXA~~.

Voir nombres BINAIRES
HEXA.
DECI

Valeur des Coeff de

16 x 4
16 x 3
+
27
= 1B -

repeat 4 [fd ou bk] 16 rt et 27 l'angle - l'inverse

(nbre de côté) la longueur longueur du côté

$2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0$
$128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255$ -

. CONTENTS

- Affiche le contenu de l'emplacement des symboles Dr LOGO

E. DEF dirpic an empty word number arctan quotient
setsplit copyoff nowatch shuffle notrace define savepic
FALSE TOPLEVEL. paddle setcursor memberp px tt recycle
st ts noformat rt ss pu rq pe pr or release lt
po op rl ht loadpic tf fs sf se uc rerandom er ct
go cs re run pe pd pps co not le if bf bk
changef erasepic int sin fd ed defaultd cur bf dot ent
pots stop lput ern text cos sety setx type pops
setscrunch show ERRACT pons TRUE fput dir > = pal list
word < wrap keyp remainder REDEFF wait last throw
seth and pprop erasefile setd item erf save / +
listp wordp plist + error * date) count home sound
(first gprop round. deposit glist load empty setpc
char make pause setpos cursor. PRM. contents. where
watch setbg. examine. APV. FMT. SPC. REM. namep erall
trace setpen . ENL. buttonp 1
thing nodes poall copyon window wenamep
thing nodes poall copyon window watch setbg . examine
APV. FMT. SPC. REM. mon namep erall trace setpen
ENL. buttonp local ascii, setpal equalp piece out,
remprop. towards catch clean erall repeat random
fence label]

Se END! TO! ED! CURSOR! PR! SHOW! DOT
FILL / SETPC / TOWARDS / NOFORMAT PONS /
END / RELEASE / BYE / LABEL / NOWATCH /
PAUSE / WATCH / INT
Les caractères / PRM / en code ASCII de 0 à
n'ont pas n'existent pas / cela berre à READY
ou replace le curseur en haut à gauche / en en bas milieu
sauf PRINT CHR$ (1) = ↵ (14 = ([illegible])
et 24 -
qui ouvre une mini fenêtre en jaune

1020 + 510 + 1530 + 255 + 2295 = 5610

16 + 16

Chiffres en BINAIRE - base de 2 de 0 à 7
de droite à gauche :

ex : $2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$
0 1 0 1 0 1 0 1

en déci : 64 + 16 + 4 + 1 = 85 -

Chiffres en HEXA - base de 16 DÉCI.

1B = (16 × 1) + 11 (valeur de B) = 27.
27 = (16 × 2) + 7 = 39
111 = (16 × 2) + (16 × 1) + (16 tout à 1 × 0) = 273 -

16 = 10. Exposeurs et non multiples -
1/2/E = 16^2 + 2
1 × 16 × 16 16^2 + (2 × 16) + 14 = 302
143 = 16^2 + 8 + 3 = 419 - (10 × 16) + 3 = 419

Charger le prog sur disc : SAVE " "
du disc à l'ordinat. LOAD " " ou RUN - " "
+RUN

0 à 9 A à F A = 10 F = 15 00FF

A 10
B 11
C 12
D 13
E 14
F 15

426191

256

20AA

20^{16} + 16 × 10 + 10 -

0101 -

16^2 0 16

(13 × 16) + C 15 3 12

00FF 7 (10 × 16) + 10
A

16

20 - (16 × 10) 10

0 1 0 0 0 0 0
$2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$ 64, bon départ
2 1
16 + 15 = 15
00FF +
16^2 (16 × 15)

20AA = 2 × (16 × 10) + 10
= 3210 -

- LOGO -

fd ou bk - longueur et (ou) largeur (avant / arrière -

rt et - rotation de l'angle - (et non la valeur

contraire en 2 traits) - (fd 30 rt fd x).

au début le nombre de côté -

ex : - (nom) to carre.

- repeat 4 [fd 50 rt 90]

- end -

(l'écran affiche le fig mémorisé ex : "carre defined)

pour le cercle -

360 [fd 1 rt 1] pour être finissé par 20 -

soit 18 mais les coordonnées x 20 ou [fd 20 rt 20] -

Une "étoile" à 9 branches -

pour être obtenue par -.

9 [fd 120 rt 160]

- le cercle / l'étoile "s'ovalisent" avec setscrunch 1.5

(0.5 etc)

ere - -

fd et RT (ou et) sont égaux ; une différence c'est que le ne joue pas sur

la forme mais sur le nombre de tours -

le rapport de fd / rt. (curieux bizzare!)

à revoir ex: 360 [fd 2 rt 1.] ?? ou 360 [fd 4 rt 1] ?

Comment effacer une disquette sans repasser par VERA.BAS ou BAK FORMATAGE ?

Comment ouvrir un fichier / le charger / et le Lire ?

42)

7FFF. F = 15.

A=10. $7 \times (16^1)(16 \times 9) + 15$ - & & & &

$(16 \times 7) + (16^2) + (16 \times 15) + 15_{16} = 639$

base 16 A= 20/AA

ZZ 7FFF - 20. $(10 \times 10$

16 +15 2. $(16^4) +$

20AA · 16 16 $^2(16 \times 10) + 10$ - $(16^3) + (16 \times 10) + 10$

$20 \times 10 + 10$

FF = 15×15 =

FRE - (expression numerique) ex: FRE (0) pour le basic

Ne pas oublier + Loin. . . en chaine alphanumerique !

HIMEM donne l'octet de la mémoire le + haut utilisé par le BASIC soit l'adresse (42619 ?) de l'octet utilisé par le basic

MEMORY Définir le quantité de mémoire disponible en fixant l'adresse de l'octet. le + élevé (MEMORY = & 20AA) ?

(RAM) = 128 Ko : en 2 de 64 Ko -
Pes CPM utiliseur Pes 128 Ko
Le BASIC seul Pes le 64 Ko. reste 64 Ko utilisable par
BANK MANAGER -
L'image d'écran = 16 Ko (bloc 1) reste 4 écrans en bloc 2, 3, 4, 5
contenus dans la 2ème partie des 64 Ko -
Le BASIC n'exploite que le bloc 0 de la mémoire (ROM ? RAM ?)
avec ? FRE("0")
? HIMEN donne l'octet le + haut de la MEM utilisé par BASIC
OCTET = 8 bits - modifiable par MEMORY (mais ne pas faire)
MEMORY adresse de la mémoire MEMORY & 20AA (RAM ou ROM) ?
PEEK lit le contenu de la case mémoire Z80
entre &00 et &FF (0 et 255) - RAM
POKE à utiliser avec précaution - inscrit la valeur dans la RAM

Revoir FONCTION SYMBOL et SYMBOL AFTER
qui avec MEMORY, HIMEN vont pouvoir redéfinir le clavier
(le modifier dans un sens irréversible) - (SYMBOL AFTER 256)
? HIMEM = 42619 ? en Octet ? = ? en déci et Hexa ?
= 20AA en Hexa ?
CALL FRE

Type de données.

Les ENTIERS ? qu'est-ce qu'un entier -
~~et les nbres réels~~ la valeur de -32768 à
32767. (= 65535 chiffre revenant souvent)
nombres réels - 1.7E+38 à + 1.7E+38
ch 3 p 31 MEX$ nombre entier sans signe
entre 32768 ? et 65535 - ?
HIMEN longueur 42619 - ? en quoi. Octet ?
MEMORY = &20AA = en déci ?

TAPER un Prgr. (sous disc). (Ce prg est en mémoire clavi.)
introduire le disc. TAPER SAVE "NOMFICH"
pour la sauvegarder sur disc. (après avoir introduit le disc bien sur)
puis CAT.

LOAD "NOM" charge dans l'ordinateur le disc
puis RUN.
ou RUN "NOM" pour démarrage immédiat.

~~Voir INKEY$ chap 3 p 34~~

Disquette = 40 pistes ÷ ~~ps~~ en 9 secteurs = 360 secteurs
1 secteur = 512 Octets ; 1 disquette = 512 x 360 = 184 Koctets ~~par face de disquette~~.

| Valeur d'un octet ? en deci / Hexa / Bin ? |

1 Octet = 8 bits / 1 bit = 0 à 256 en decimale ?
1 Kylo Octet ou KO = 1024 octets (et non 1000)

47262. 159. (&9F)
~~47295~~ - 47320 ~~47337~~ à ~~48714~~
à ~~48714~~ - ~~48755~~
~~49022~~ ~~49156~~
0 de ~~48715~~ à ~~48755~~
~~49155~~ ?
~~49177~~
0, ~~48775~~
~~49196~~ 49215

0 de 49216 à 49351

de 49352 : 112
à ~~49374~~ 48
~~49405~~ ~~49438~~

0 de 49433 à 49500 et + (stop)

POUR [METTRE] sur DISC un prog -
- taper le prog sur le clavier -
- puis introduire la Disquette -
- Taper SAVE "exemple" [RETURN] + BAS (ou BIN)
- Pour le [charger] ou CLAV par + de 8 lettres -
dans l'ORDINATEUR -
- Introduire la disquette.
Taper LOAD "Exemple"
- puis RUN.
ou RUN "exemple" pour un demarrage immediat.
Les progs s'enregistrent les uns apres les autres sans s'effacer -

---

POUR CONNAITRE le MESSAGE D'ERREUR plus precisement: Taper PRINT DERR
et voir chap 7 p 27. et p 31

Pour verifier le CODE de CONTROLE de la CONSOLE Tapez en CP/M 1. pour 1.
apres A> : REN TROFILE.SUB=~~REN~~ TROFILE.ENG
chap. 4 page 3 - ~~LANGUAGE 3~~
SETKEYS KEYS.CCP
~~LANGUAGE 3~~
et le petit code controle : Chap. 5 page 19.

LOGO

Fonction ?PONS il faut qu'un dessin soit definie pour obtenir la cote - ou autre parametre -

Listing coucher et lever du soleil.

220 - 1 ou I ou l ? (l - pour la 1ère tentative)

240 idem.

250 idem 580 / 890 / 980 / 1260

Rappel du programme taper sur

DISC SAVE "JOUR" (

puis - LIST.

En BAK.

LOAD "JOUR.BAK"

RUN

LIST

~~10 (q ou g (?) (à la fin de ligne)~~

aussi 90 - 150 - 170 - 250 590 600

~~Adresse mémoire Contenu Mémoire~~

~~42904 &A798 255 &FF~~

~~42912~~

~~42928 42952 42960 42975 43005~~

~~44418 44460 45315 - à 455~~

~~(45316 à 45544 0~~

~~(45545 &B1E9 6 (&6)~~

~~à 45546 &B1EA 83 (&53)~~

~~45559~~

~~de 45547 à 46229 0~~

~~45560~~

~~(de 46236 (&B494 240 (&F0)~~

~~à 46509. 13~~

de 46510 à 46631 = 0

de 46632 à ~~46664~~ ~~46755~~ = ~~255~~ ~~46900~~ 47155

0 de ~~47129~~ à de 47156 à 47261

24 195
25. 199 -
26. 185 -
27. 195 -
28. 185 -

PEEK - chap. 3 p 56 -

Adresses Memoire Vive RAM -

	Adresse	Mémoire	Contenu	Mémoire
de	0	(&0)	1	(&1)
à	96 72	(&60)	34	(&22)
de	97 -	(&61)	0	(&0)
à	367	(&16F)	0	(&0)
de	368	(&170)	12	(&C) -
à	597 613	(&254)	4	(&4) -
de	597.613	(&255)	0	(&DO - (&4) (&0)
à	12000			

stopp à 12000 -

temps de 0 à 10000 = 43'20" suite 2 pages

decalage sous Mémoire à partir de 10000 - + loin -

temps pour 1000 à partir de 10000

7'40" soit pour arriver à 65535 -

43'20" + 7h 27'20" + 3'50 = 8h 15' ≃

de 0 à 368 Système ou 8000 à &0170

368 à 43903 Basic ou &0171 à &AB7F

43904 à 49151 Système ou &AB80 à &BFFF

4096 à 65535 Memoire Ecran &C000 à &FFFF

CONVERSIONS

des chiffres

avec la calculatrice

Le reste d'une division se retrouve en retranchant des nombres avant la décimale (nombre entier) du résultat, et en x par le diviseur

ex : 789 : 12 = 65,75

retranche 65 = 0,75 166 Oct F0

reste 0,75 x 12 = 9 123 Oct F1

&H AG 7B

de décimal à Binaire

Chercher par tâtonnement le nombre de puissance de 2 que contient le chiffre + Noter ce nombre faire idem avec le reste

ex : pour 173 Déci

il y a $2^7 = 128$ reste 45

$2^5 = 32$ " 13

$2^3 = 8$ " 5

$2^2 = 4$ " 1

$2^0 = 1$

soit

7	6	5	4	3	2	1	0
1	0	1	0	1	1	0	1

VOIR EVERY REMAIN DEC$ STR$

Calcul de l'Octet

- l'Octet = 8 bits en binaire soit 11111111 = 255 max
il se décompose en 2 quartets de 4 bits
qui comporte l'Octet Fort que l'on trouve en 1er
l'Octet faible en second (ce n'est pas le quartet
du chiffre qui est importante en Octet faible pour avoir
un chiffre + élevé qu'en Octet fort

~

Soit le chiffre déci 34065 ÷ 256 = 133
Octet Fort - &8521 du Déci à l'Hexa
Reste 17 Octet faible &11.

37315 : 256 = 145 Octet Fort reste 195
l'Octet faible -

- ~~256~~ 145 = &91 en Hexa -) que l'on trouve
195 = &C3 " ") en ÷ par 16
et reste -

Soit 37315 = &91C3 -
déci Hexa -

133/1 -

du Binaire à l'Hexa -

&X 1101 1001 . = 217 = &D9 -

quartets - 1101 1001
déci 8+4+1 8+1
= 13 = 9
Hexa. = &D = &9
= &D9

3 2 1 0
&8511 -

— suite Adresses —

42527 -

Adresse	Mémoire Vive RAM -

~~42616~~ & 65
~~42904~~ & 798 255 & FF
~~49432~~ 52400 step to 48

0 (de 49434 à 49553 -
(49542 & ~~C192~~ 112 ~~16~~ & 10

50090 ≈ . 50153 50176 50203 233
50313 336 363 392 / 477 / 496
523 / [illegible] / 637 / 656 / 683 / 712 / 796
50817 / [illegible] / 872 / 952 / 976 / 51008
51032 / 116 / 132 / [illegible] / 244 / 400 / 423 / 451 /
51481 / 56[illegible] / [illegible] / 611 / 64[illegible] / [illegible] / 743 / 771 / 801 / 881 / 905
51931 / 961 / 042 / 066 / 09[illegible] / 121 / 201 / 225 / 251 / 281 / 361 / 385
52411 / 441 / 526 / 545 / 571 / 601 / 683 / 705 / 731 / 761 / 843 / 865 / 891 / 921
53008 / 025 / 54[illegible] / 63 / 185 / 277 / 315 / 449 / 472 / 499 / 529 / 610 / 681
53659 / 682 / 769 / 791 / 819 / 849 / 929 / 953 / 979 / 54009 /
54089 / 93 / 111 / 139 / 170 / 249 / 273 / 300 / 330 / 410 / 434 / 460 / 490 / 573 /
54594 / 620 / 650 / [illegible] / 753 / 778 / 810 / 893 / 913 / 940 / 970 / ~~55[illegible]~~ 55052 /
55074 / 100 / 130 / 210 / 232 / 269 / [illegible]98 / 520 / [illegible] / 577 / 657 / 682 / 708 / 738 / 818
55840 / 867 / 897 / 970 / 56000 / 27 / 85 / 136 / 160 / 188 / [illegible]8 / 299 / 322 / 347 / 377
56457 / 482 / 508 / 537 / 621 / 641 / 668 / [illegible]98 / 780 / [illegible] / 830 / 860 / 940 / 960 / 990 / 57018
57102 / 20 / 147 / 78 / 260 / 280 / 460 / 540 / 560 / 596 / 625 / 705 / 730 / 760 / 785 / 865 / 888
57915 / 945 / 025 / 050 / 076 / 106 / 185 / [illegible] / 236 / 265 / 345 / 370 / 396 / 426 / 506 / 530
58555 / 587 / 690 / 715 / 745 / 830 / [illegible] / 875 / 905 / 985 / 010 / 035 / 066 / 148 / 170 / 195
59225 / 305 / 327 / 460 / 594 / 616 / 644 / 673 / 753 / 775 / 804 / 833 / 913 / 936 / 965 / 994
60074 / 98 / 124 / 154 / 233 / 257 / 284 / 314 / 394 / 418 / 444 / 474 / 554 / 577 / 603 / 633
60738 / 64 / 94 / 877 / 897 / 923 / 953 / 036 / 58 / 84 / 113 / 61196 / 218 / 243 / 274 / 356 / 378
61508 / 642 / 663 / 692 / 721 / 801 / 823 / 852 / 881 / 962 / 983 / 11 / 41 / 122 / 143 / 171 / 201 / 28
62302 / 332 / 360 / 441 / 465 / 492 / 531 / 600 / 625 / 651 / 680 / 766 / 784 / 811 / 841 /
62925 / 944 / 972 / 63000 / 22 / 85 / 105 / 130 / 160 / 240 / 265 / 292 / 321 / 404 / 425 /
Vide de 63425 à 65535

Complet 42616 à 49554 ~

Complète de 63425

Vide de 63425 à 65535 (Mémoire Écran)

RAM : Complète de Ø à 613 / soit &Ø à &265
Complète de 42616 à 63425 soit &A678 à &F7C1
Util. Cassette de 1000 à 40.000 = &3E8 à &9C4Ø -
avec HIMEM/MEMORY = &A67B = 42619

- Comment se sert-on des fonctions, ↑ - le MEMORY
- STOP pour vérifier (ou tester) les différents modules ne change RIEM
d'un programme ?
- d'ON ERROR GOTO pour intercepter les entrées erronées ?
- ~~Que signifie la valeur~~ en code ASCII
d'un caractère quelconque ~~du clavier~~ obtenu
~~par ?~~ ASC ("x") - ~~un défaut~~ avec le
~~tableau "Valeurs ASCII par défaut (2ème page du Manuel)~~ ?
~~ou chap. 7 - p21 à 22~~ ? C2/8 3/14.
- Comment placer le curseur en un point quelconque
de l'écran (Locate ne semble pas y aller)
hSHOO=0 ~ 1000 3/232
Pour le prg "Livre BASIC M.1" E8
- Message : il faut mettre un GOTO 40 pour
que le défilement se poursuive sans discontinuité -
mais s'il faut changer ces 4 en 2 pour mode 0
Mais Cassidy la ligne 110 LOCATE 40-i,1
un dédoublement du message !! et autres
ne fonctionne pas en mode 2 (vu)

Vérifier Z et X
et les fonctions
Code

METTRE au propre
et chercher

~~Valeurs~~ des Touches en Déci :

× [A] = 1/65	[V] = 22/86	× a = 97	. v = 118		
[B] = 2/66	[W] = 23/87 ?	b = 98	w = 119		
[C] = 3/67	[X] = 24/88	c = 99	x = 120		
[D] = 4/68	[Y] = 25/89	d = 100	y = 121		
[E] = 5/69	[Z] = 26/90 ×	e = 101	× z = 122		
[F] = 6/70		f = 102			
[G] = 7/71	[↵] = 167 ?	g = 103 ?			
[H] = 8/72		h = 104			
[I] = 9/73	[CLR] 127 ou 16	i = 105			
[J] = 10/74		j = 106			
[K] = 11/75	[↑] 248 – 244 – 240 –	k = 107			
[L] = 12/76		l = 108			
[M] = 13/77	[←] 244 246 246	m = 109			
N = 14/78		n = 110			
O = 15/79	[→] 251 247 243	o = 111			
P = 16/80		p = 112			
Q = 17/81	[↓] 249 245 241	q = 113			
R = 18/82	[DEL] = 127	r = 114			
S = 19/83		s = 115			
T = 20/84		t = 116			
U = 21/85		u = 117			

Chiffres en haut du clavier

× 1 = 49
2 = 50
3 = 51
4 = 52
5 = 53
6 = 54
7 = 55
8 = 56
9 = 57
× 0 = 48

seul le 2e chiffre est valable ?

si possible Cas n° des autres fonctions

Classer par fonction -

Chiffres du pavé numérique

& ou ! = 33 x	: = 58 x	0 = 128
" = 34	; = 59	1 = 129
# = 35	< = 60	2 = 130
$ = 36	= = 61	3 = 131
% = 37	> = 62	4 = 132
& = 38	? = 63	5 = 133
' = 39	à = 64 ?	6 = 134
(= 40	[= 91 x	7 = 135
) = 41	\ = 92	8 = 136
* = 42 x	] = 93	9 = 137
+ = 43	^ = 94	. = 138
, = 44	_ = 95 ?	ENTER = ~~139~~ 13 ? / ~~13~~ ?
- = 45	` = 96 ?	TAB = 9 / 45 ?
. = 46	è = 123 x	RETURN = 13 &
/ = 47 x	ù = 124 x	COPY = 224 ? attention
0 = 48	é = 125 x	touche d'espacement
1 = 49	= 126	ou = 32 ? attention
2 = 50	DEL = 127	reste SHIFT = 244 ?
		CONTROL
		CAPS LOCK
		ESC

32 caractères d'extension

de 128 à 159 Clés de 0, à 30

dont 128 à 140 les chiffres du pavé NUMERIQUE

0 à 9 le point et ENTER

- VARIABLES -

Caractère ou groupe de caractères qui représente une valeur - Elle peut être reelle / entière / de texte - doit toujours commencer par une lettre

reelle : toutes valeurs entre +/- 1.7 E 38 . (le point à la place de la ,)

entiere " " +/- 32767

de texte comme valeur symbole de 0 à 255 entre " "

avec $ elle est "de chaine"

avec x(0), x(1), x(2) .... elle est indexée

voir [!] fonction

ON ERROR GOTO ?

Tabulation ?

espace entre " et texte ou variable n'importe pour la lecture

Pour les Operations + - x ÷

on compte par comp. n'impliquant que 2 (ou 3, 4 etc) .

10 INPUT : "1er Nombre" ; a

20 INPUT : "2e Nomb" ; b

30 Resultat ; a + b (ou a - b ou a * b)

Pour la division / obstacle divisée par 0

est impossible alors 10 INPUT "1er nombre" ; a

20 INPUT "2e Nombre" ; b / 30 IF b = 0 then &0 else 50 / 40 ? "impossible" : GOTO 10 /

50 ? "Resultat =" a/b / 70 GOTO 10

MATH

Calcul des LOG 10 à la chaîne

~~10 INPUT~~ 10 FOR n = 1 to 20
20 PRINT LOG10 (n);
30 NEXT n

Pour le pas STEP sur la ligne 10

(ex 10 FOR ~~x~~ n = 0.001 to 1 ~~Step 0.001~~ point → 2)

idem 10 ~~PRINT~~ ~~LOG10~~ (10

10 FOR n = 0,1 to 1 Step 0.1

idem pour les LOG NEP

mais 20 PRINT LOG (n) (chez de 10)

Pour les arc tangentes idem (mettre DEG en ligne 5 #)

Pour les ~~ARC~~ / ~~COSINUS~~ COTANGENTE

20 PRINT 1/TAN (n);

autre système chiffre par chiffre

~~1~~ 5 REM (multiplication)
10 INPUT " 1er nombre "; a
20 INPUT " 2 ème "; b
30 ' Calcul ou comp pour calcul

10 REM
20 INPUT " 1er Nombre "; a
30 INPUT " 2ème Nombre "; b
40 PRINT " Resultat "; a * b (pour la multiplication)
50 GOTO

Se complique avec la division
car si b = 0 est celle qui est impossible

Boucle de temporisation :

Fait tourner le µ pendant un temps donné

```
10 FOR X = 1 TO 50 } 1 seconde
20 NEXT X
```

- le µ ayant une vitesse d'exécution de 0,02 seconde x par autant de fois 50, pour qu'il tourne à vide - avant une autre instruction.

~

—

Pour les fonctions déjà en MÉMOIRE

LOG10 / LOG / SIN / COS / TAN ATN -

```
10 INPUT "nombre" ; a
20 PRINT "LOG DECI de" ; a ; "="
30 PRINT LOG10 (a)
40 GOTO 10
```

pour LOG Nep. taper 30 PRINT LOG (a).

~

pour SINUS / COS / TAN / ATN avec préciser en début DEG (= degré) sinon Radian !

3
EX ADD

~

ASSEMBLEUR Espacements ligne N° 216

"LD" 2/2/1/2/1/1/1/1/2/1/2/2/1/1/1/1/1/1
2/2/1/1/2/2/1/1/1 * * 8 / * 1/1/2/1"

1 sur X

ASSEMBLEUR Espacement

"LD/2/2/1/2/1/1/1/1/2/1/2/2/1/1/1/1/1/
1/2/2/1/1/2/2/1/1/***/1/1/1/2/1"

sur ASS : "LD/2/2/1/2/1/1/1/1/2/1/2/2/1/
1/1/1/1/1/2/2/1/1/2/2/1/1/***/1/1/1/1/1"

Le prog BANKMAN chap 9 p 56

Saver l'ensemble sur disquette

"JOURNOIT Face 1 sous le nom de FICHIER

et Face 2 ~~sous le nom de BANK et BUK~~

Une correction des lignes 1720 (une 9 au lieu de :) et pour "BUK" 2660. La fin de cette ligne se termine par ~~?~~ qui se transforme en PRINT - ? + Face 1. 1730 tempx + X

Ligne 110 "XOR" mise en minuscule

Ligne 280 entre ligne 2470 ; X ; Y ; : PRINT #1 "MODE TRACE

2740. CLS : GOSUB : " PRINT " Erreur survenue à la ligne" ;

2745 PRINT : ERL : PRINT : LIST

ASSEMBLEUR "LD 2 JR 2 DJNZ CALL RET 1 JP 2 INC 1 DEC 1 PUSH POP 1 RST 1 IN 2 OUT 1 IM 2 EX 2 ADD 1 ADC 1 SUB 1 SBC 1 AND 1 XOR 1 OR 2 CP 2 RLC 1 RRC 1 RL 2 RR 2 SLA 1 SRA 1 ?? ?1 SRL 1 BIT 1 RES 1 SET 1"

42620 = &A6FC

~~&H~~ 166 124 = A6 / 7C = &A67C

- RAM / ROM -

- Le CPC 6128 possède 128 Ko de RAM -
- mais seuls 64 Ko sont exploitables directement -

Sur ces 64 Ko de RAM -
sont divisés en 4 blocs de 16 Ko -
numérotés de 0 à 3 -

bloc 0 : Basic système d'exploitation / RAM et ROM /

bloc 1 &H4000 = 16384 = BASIC / RAM /

bloc 2 = &H8000 = 32768 / Données (Basic, Basic etc) variables
(ou &7FFF critique)

bloc 3 = &HC000 = 49152 à &HFFFF 65535
ECRAN + INTERPRETEUR BASIC + ROM disquette

~

Plus d'écran après &7FFF (ou 32767) -

- RAM &000 à &0170 (0 à 368) système
369 à ~~42620~~ 42620 Pour programmes Basic 43903
(&0171 à AB7F)
&A67C (?) à BFFF système (42620 à 49151)
&C000 à &FFFF (49152 à 65535) mémoire Ecran
ROM &000 à 3FFF (0 à 16383) système
&C000 à &FFFF (4096 à 65535) Basic
caractères &3800 à 3FFF (2048 à 16383)
Mots instructions 58248

Pour le prog " Message défilant " - ADD / ADC
"Livre Basic" (MA) page 74 -   SUB / SBC

pour que le message se déplace droite à gauche intervertir 80 et 120 ainsi que 70 et 110 -

- Pour prog ch 1 page 56 voir BASIC (Livre) pour défilement de haut -> bas etc page 110 -   chap. 7 p 46

+ Les caractères définis par l'utilisateur se trouvent immédiatement au dessus de la limite HIMEM, mais celle ci peut être modifiée par MEMORY et elle est automatiquement abaissée de 4 Ko lors de ~~l'ouvertu~~ l'ouverture de fichiers AMSDOS pour création d'une mémoire tampon + le nombre de caractère défini par l'utilisateur ne peut être modifié que si aucun changement de prog de caractère

ASSEMBLEUR — le recherche dans le listing

donne des resultats erronés sur ABC ~~B~~, — de A à H

~~ADC~~

LD, ABC, A, 0   FAUX

CP - B = BA (non BB) B8   CP. B

CP - C = BB non BC) B9   CP. C

CP - D = B A (exact)

CP - E = BB (exact)

CP H = BE non faux) BC

CP L = BF non faux = BD

~~CP H ?~~

~~CP A H ?~~

ADD a, b = 82 Faux

ADD a, c = 83 Faux

~~ADD a, d~~ = 86 Faux

ADD a, H = 87 Faux -

ADD a, L = 8A Faux

ADC, a, b = 8B Faux

ADC, a, c = 8E Faux

ADC, a, H = 8F Faux

ADC, a, L = 92 Faux

SUB a, b = 93 Faux

SUB a, c = 96 Faux

SUB a, h = 97 Faux

SUB a, L

arc/sin/cos/cotg-

FONCTIONS TRIGOS

arc/cosinus

```
5 DEG (si nécessaire)
10 INPUT "Cosinus" ; X
20 Y = -ATN(X/SQR
     (-X*X+1)) + 90
30 PRINT Y
```

SOA B = CB2A C = CB2B H = CB2E L = CB2F

arc/sinus

```
5 - DEG (si nécessaire)
10 INPUT "SINUS" ; X
20 Y = ATN (X/SQR(-X*X+1))
30 PRINT Y
```

SLA B = CB22 C = CB23 H = CB26 L = CB24

0 à précision jusqu'à 1,5 ensuite valeur fausse

cotangente

```
5 - DEG (si nécessaire)
10 INPUT "Cotangente" ; X
20 Y = 1/TAN(X)
30 PRINT Y
```

RR B = CB1A C = CB1B H = CB1E L = CB1F

Pe 1.5708 donne pour les RADIANS
+ précis 1.570963
(doit R/2)-

ASSEMBLEUR – 186 = ou

CP a
Break 10390

Fausses toutes les données

CP	B = BA =	ADC	a, B = 8A
"	C = BB	"	, C = 8B
"	H = BE	"	8E
"	L = BF	"	H = 8F
			L =
ADD	a, B = 82	SUB	a, B = 92
"	C = 83	"	a, C = 93
"	H = 86	"	a, h = 96
"	L = 87		a, l = 97 ?

SBC – a, b = 9A	AND A,B A and B = A2
SBC a, C = 9B	A and C = A3
" a, H = 9E	A and H = A6
a, L = 9F	A and L = A7
XOR a XOR B = AA	OR A OR B = B2
A XOR C = AB	A OR C = B3
A XOR H = AE	A OR H = B6
A XOR L = AF	A OR L = B7

~~INC et DEC~~ RLC B = CB07 faux
C = CB03 (f)
H = CB06 (faux
L = 6B07 (faux

RRC B = CB0A
C = CB0B
H = CB0E
L = CB0F
) Faux.

arc tangente 5 DEG

```
10 INPUT "Tangente"; x
20 Y = ATN (x)
30 PRINT
```

arc cotangente 5 DEG

```
10 INPUT " cotangente"; x
20 Y = 90 – (ATN (x))
30 PRINT Y
```

il y a inversion ou répétition des résultats entre

A, A (juste)
A, L (faux)
A, B faux
A, D (juste)
A, H faux
et A, L juste
A, L faux
A, A juste

RL
B = CB12
C = CB13
H = CB16
L = CB17
) Faux

ADDRESSAGES -

- IMMEDIAT ex : LD A, n - charge n dans l'Accu de façon immédiate / ADD A, n ajoute n à l'Accu
  " " "
  HL Pointeur de données principal -
  BC et DE " auxilliaire -
- REGISTRE ex : LD A, L (entre registres) -
  (ou appelle inhérent ou implicite).
  Registre d'Instruction (PC) = (16bits) contient l'adresse en cours / Lorsque le prgr est lancé le PC (Compteur Programme) est fixé sur la 1ère case du prgr / Après le prgr accomplis le PC est mis à jour et pointe sur l'instruction suivante.

Il y a des sauts JP - inconditionnels / et conditionnels JR

- DIRECT - Parce qu'il utilise le contenu des cases mémoires ou des registres directement, c'est à dire sans en faire quelque chose auparavant :
  ex : LD (2000), BC.
  recopiera le contenu du Registre double BC, dans la case mémoire 2000. Noter les parenthèses.
  L'inverse LD BC (2000) chargera le contenu de la ~~case mémoire 2000 dans BC~~ BC les cases 2000 et 2001
- Concerne une adresse et un registre (simple ou double)
  dans l'ordre l'ex: ~~LD (2000)~~
  LD BC, (2000)
  C recevra l'OF4 de 2000 et B l'OF0 de 2001 -

Les parenthèses (x x) qui entourent une chiffre, ou un groupe de chiffres et/ou lettres indiquent qu'il s'agit de la case mémoire pointée par ces chiffres et ou lettres, qui est concernée - Ex: LD A,(2000) c'est la case mémoire n°2000 son contenu qui est chargé dans l'Accu + De même LD A(HL)

INDIRECT. ex. LD r, (HL) chargera dans le registre r le contenu de la case mémoire pointée par HL -

LD (HL)r chargera dans la case mémoire pointée par HL le contenu du registre r.

LD A, (BC) chargera dans l'accumulateur le contenu de la case mémoire pointée par BC.

LD A, (DE) " " pointée par DE.

+ Concerne uniquement des registres, simples ou doubles -

- + Différence entre DIRECT et INDIRECT, le 1er contient une adresse pas l'autre + Le 2ème (INDIRECT) ne contient que des registres -

- Analogie + Ils utilisent les parenthèses + Mais dans les 2 cas, ils copient ou lisent le contenu des adresses ou registres pointés. (et non les registres ou adresses) -

INDEXÉ ex: LD A, (IX + 3)

Chargera dans l'Accumulateur le contenu de la case mémoire pointée par le registre IX (d'index) plus 3 - + Supposons IX pointant l'adresse 200

LD A, (IX + 3) :

	Adresse	Contenu
LD A, (IX + 3)	200	1
	201	2
	202	3
LD A, (200 + 3)	203	4

A = 4 et non 3 !

ou tout autre registre double
suivit d'adresse (&XXXX)

- A la fin de chaque listing toutes les valeurs doivent être chargées dans l'A pour l'écrire à l'écran (BBSA)

DIFFERENCE entre LIRE et ~~CHARGER~~ → ECRIRE

- Ex: LD A, (~~&BF93~~ 35000) (35000)

L'instruction LD lira le contenu de l'adresse ~~&BF93~~ 35000 et le chargera dans l'A - (ou l'écrira) -

(Vérifier avec PEEK la valeur de l'adresse)

- Ex: LD (~~&BF93~~ 35000), A -

L'instruction LD écrira à l'adresse 35000 ~~&BF93~~ la contenu de l'A (accumulateur) - (si la valeur ne dépasse pas 255).

~~&BF93~~ peut être pris comme exemple (sa valeur = chiffre variable)

Ex:
```
10 ENT.
20 LD DE, 65      (65 = Asci de "A")
30 LD (35000), DE (65 est écrit à l'adresse 35000)
40 LD A, (35000)  et chg écrit dans Acc
50 CALL 47962     (Affiche à l'écran "A")
60 RET.
```

~

DEC diminue de 1 -
INC augmente de 1 -

~

ROTATION et DECALAGE -

Décalage : Déplace (vers la gauche ou droite) le contenu d'un octet, bit par bit. Le bit "expulsé" est placé dans le Carry. Le bit entrant est Ø

Rotation : Déplace (vers la gauche/ou droite) le contenu d'un octet, mais au contraire du décalage
- le bit entrant est soit : - le bit "expulsé" soit le bit Carry -

~

+ Décalage : SLA ← gauche / Bit expulsé dans Carry -
bit entrant = Ø -
- SRA → droite / bit expulsé dans Carry / bit entrant !
- inchangé ! (si bit 7 = bit de Signe) -
SRL → droite / bit expulsé dans Carry / bit entrant = 0 -

~

+ Rotation : RL ← gauche / bit sortant dans Carry / bit entrant "ancien Carry -
- RR → droite idem mais vers la droite -
- RLA ← gauche / bit sortant dans le Carry de l'Acc / bit entrant "ancien Carry - RLA et RRA spécifique à l'Acc
- RRA → droite idem, mais vers droite -
- RR (HL) ← gauche, rotation d'une case mémoire pointée

par (HL) à travers le Carry + Bit sortant dans le Carry + bit entrant "ancien Carry"

- RL (XY+d) idem sur case mémoire indexée
- RR (HL) → droite/rotation d'une case mémoire pointée par (HL) à travers le Carry.
- RR (XY+d) idem sur case mémoire indexée.

En somme, les rotations ci-dessus s'effectuent sur 9 bits + Le Carry étant le 9ème bit qui participe à cette rotation.

~

- RLC ← gauche / il s'agit d'une rotation vers la gauche, mais sur 8 bits seulement. En effet le bit n°7 "expulsé" va prendre la place dans bit n°0 (Une + copie du bit n°7 va dans le Carry)
- RRC → droite / idem / mais le bit n°0 expulsé va prendre la place du bit n°7 - (Une copie du bit 0 dans le Carry)
- RRCA → droite mais spécifique à l'Accu.

- RLCA ← gauche rotation à gauche spécifique à l'Accu + Mais sur 8 bits de l'opérande seulement le bit n°7 "expulsé va prendre la place du bit n°0

MULTIPLICATION

En fait, les rotations ci-avant s'effectuent sur 8 bits, le bit n° 7 "expulsé" dans le cas de RL va prendre la place du bit n° 0, qui devient bit n° 1.
Dans le cas de RR, le bit n° 0 expulsé va prendre la place du bit n° 7, qui devient bit n° 6.

En somme les rotations s'effectuent sur les 8 bits de l'opérande.
Mais dans le cas de RL... (gauche) si le bit n° 7 est égale à 1 il va prendre la place n° 0 (ou n° 8) dans l'octet à gauche ou

Ex : LD B, 128 soit 10000000
RLC (espace) B 00000001
LD A, 65. 01000001
ADD A, C. 01000010 = 66.
CALL &BB5A. Lecture à l'écran
RET. / 66. soit l'ASCII = "B" chr$(66) = "B"

dans le registre FLAG. où sa valeur est de 1.
Il se trouve ajouter à l'Accu sur l'exemple ci-dessus.
L'exemple ci-dessus n'était pas un bon exemple puisque que bit 7 (bit signe) soit un nombre négatif. (-128)

Dans une Multiplication ~~16~~ bits, C = Multiplicande
E = Multiplicateur, D contiendra une partie (Octet Fort) du ~~Total en cours~~, "produit partiel invisible" (DE en fait). Comment, où (quelle position) se retrouvera le Carry ou la retenue lorsque à force de décalages (SLA E) vers la gauche de E, celui ci (Carry ou retenue) dans le registre D ?
(En quelle position dans l'Octet D ? ~~0 à 7 ??~~ -
il faut RL D pour le ~~sortir~~ du Carry soit 2° = 1 en D
HL contiendra le "Total en cours" (ADD HL, DE)

Pour le Carry : La Multiplication 2 x 128
2 ou 0,10 se trouve en C.
128 ou 10000000 se trouve en E. Lors du décalage de E le bit 7 (qui est = à 1) tombe dans le Carry (décalage SLA E) et ce n'est qu'avec l'Instruction RL D (Rotation gauche de D) qu'il prend place dans l'Octet D à la position bit n° 0. (Sans Rotation, si ce n'est du Carry à l'Octet D)

Ex : LD C, 2 (Multiplicande)
LD B, 8 (Compteur)
LD E, 128 (Multiplicateur)

LD D,0 -
LD HL,0
MULT : SRL C = C = 01 (Carry = 0) -
NOADD : SLA E E = 0 (bit 7 expulsé Carry = 1) -
RL D D = 00000001 (recharger le Carry dans D)
~~Mais~~ SRL C C = 0 (Carry = 1) -
ADD HL,DE soit HL = HL + DE -
HL = 0 -
D = 00000001
E = 00000000 -
DE = 00000001 00000000
D E
soit HL (=0) + DE (= 2^8) = 256 -

En Langage Machine - les autres, instructions changements etc - vont toujours de la SOURCE vers l'Objet : SOURCE → OBJET.

Instruction : LD A, 20 ; charger 20 dans l'A -

0 ← 8

Code - LD (Objet) A, 20 = 3E (Source) 14 - après assemblage

DIVISION

16 bits par 8 bits -
Charger le Dividende dans HL
" le Diviseur dans C (8 bits)
Compteur (8 bits) dans B -

NXT : Pour décaler à gauche les 16 bits du Dividende
l'opérer sur lui même ADD HL, HL
LD A, H - Charger l'octet du Dividende dans l'A
SUB C Soustraire Diviseur
JR C, NXTB : Si Carry = 1 sauter NXTB :
LD H, A Recharger l'octet du Dividende.
INC L Incrémenter (+1) la réponse
NXTB : DEC B Décrémenter Compteur
JR NZ, NXT : Resauter à NXT si NZ (reste des bits ÷)
LD A, L - Charger la réponse (quotient) dans l'A
CALL &BB5A Afficher
RET Retour -

Valable pour 8 bits -

NOMBRES

ENTIERS de -32768 à +32767 avec %
DECIMAUX (ou FLOTTANTS) ou RÉELS - avec !
de +1.7E+38 à -1.7E+38 -

- LECTURE et ECRITURE -

LECTURE: Lorsque l'on charge le contenu d'une adresse, ou le contenu d'une adresse pointée par 1 registre, simple ou double → dans 1 autre registre simple ou double ou indexé -

Ex: LD B,((A000)) contenu de A000 → B :
LD C,((HL)) contenu pointé par HL → C -
LD D,((IX+d)) contenu pointé par (IX+d) → D -
LD DE,((A000)) contenu de A000 → DE -

- LECTURE : JP ey = Lecture - lorsque le contenu d'une adresse, (entre parenthèses), ou le contenu d'un registre simple, double ou indexé, (entre parenthèses) va dans 1 registre simple, double ou indexé - -

ECRITURE Lorsque le contenu d'un registre, simple ou double, → va s'écrire dans la mémoire à une adresse pointée, directement, ou pointée

à l'adresse d'un registre double, simple, indexé.
Ex: LD (A000), A - contenu de A → (A000)
~~LD (HL), B - contenu de B → (HL)~~
~~LD~~
Ex: LD (HL), B - contenu de B → (HL)
LD (IX + d), C contenu de C → (IX + d).

ECRITURE - Lorsque le contenu d'une valeur 8 bits va s'écrire dans la mémoire à l'adresse pointée par un registre double, simple, indexé -
Ex: LD (HL), &FF - FF va charger à l'adresse pointée par (HL) -
Lorsque le contenu d'une valeur 16 bits va s'écrire dans la mémoire aux adresses adr et adr+1 -
Ex: LD (A000), HL - (~~HL = AB80~~)
OFA dans L, dans A000 -
OBO dans H dans A001 -

ECRITURE - Lorsque le contenu d'un registre va s'écrire dans une adresse directe, ou pointée, qui se trouve entre parenthèses).
Lorsque le contenu d'une valeur 8 ou 16 bits

va s'écrire à une adresse pointée par un registre simple ou double qui se trouve (entre parenthèses)

PSEUDO - INSTRUCTIONS

- ENT = Suivit d'un nombre Hexa / ou Déci.
(Suivant les Assemblers) il indique le point d'Entrée. (Soit ENT ou ENT $ ou ENT @ 7000).
- Le point d'Entrée est soit fixé par défaut, (Assembleur AUTOFORMATION = ENT) = 8CA0 = 36000.
- de même pour le DEVPAC soit vous fixez l'adresse de GENA à 1000 - GENA partira à 25AD - soit - 9645 - avec ENT $ -
Soit fixez par vous même - les deux :
ENT # 7000 = le Point d'Entrée -
ORG # 7000 = début du programme -
- Votre programme commencera à # 7000 -

Pour l'Assembleur "AUTOFORMATION le point d'entrée par défaut est @ 8CA0 = 36000 (ENT.) - avec ENT @ A000 + ORG @ A000 il sera @ A000.
Pour GENA chargé à 1000 -
le point d'Entrée par défaut est @ 25AD = 96#5 (ENT $)
avec ENT # A000 + ORG # A000 il sera de A000.

EQU : signifie = / Ex PRINT : EQU #BB5A -
(CALL PRINT = CALL #BB5A).

Les DEF

DEFS - suivi de 1 à N - Réserve, ou alloue une zone de mémoire de 1 à N octets de grandeur - (grandeur) - Ex : DEFS 2 = 2 octets -

DEFB - suivi de 1 à N - Place la valeur ci-dessus, à une adresse précise en mémoire - Si l'adresse en face de DEFB est 1 A000 par exemple, et que la pseudo instruction soit DEFB 2 ; la valeur (ou adresse, ou chaine) sera stockée en 1 A000 et 1 A001 + La chaine doit se terminer par DEFB 0, ou 0 qui indique la fin du mot à sortir.

DEFW - permet de placer une valeur à une adresse précise en mémoire, mais sur 2 octets - (16 bits) - à cette différence près, elle est identique à DEFB - (qui ne place que 1 (x N) octet à une adresse /

Ex : ADR : DEFW # 95FF définir une adresse sur 2 octets -

DEFM - Permet d'installer une chaine "Ascii (de 1 à 255 caractères) + Il convient d'allouer, de réserver une zone de mémoire avec DEFS -

Ex : TEXT : DEFM "ABCDEFGHIJKL 12345678"
FINTXT : DEFS

pour à PAUSE -

```
Ex : 5  ORG 40000
10  LD BC, 40009      <- BC = Début Table DEFW
20  LD HL, 40020      <- HL = (Fin TABLE DEFS -
                          Début des 4 octets réservés
30  JP &BCD1          <- Routine pour RSX -
(30 bis ou CALL &BCD1) -
40  DEFW &40014       <- Adresse des 1er DEFB (ou DEFM)
                          Saut aux PARAMETRES
50  JP 40024          <- JP 40024 -
60  DEFB &50,&41,&55,&53   Code Ascii de P, A, U, S.
(ou 60 DEFM "PAUS")        idem mais "Chaine) -
70  DEFB &C5          <- Code Ascii de "E" + &80 -
(ou 70) DEFB "E" + 128)    idem "E" + &80 -
80  DEFB 0            <- Fin des Lettres
90  DEFS &04          <- Fin de Table - Nombre
                         d'octet pour PAUSE
                         = 4 -
100 (Paramètres) -
(bis 100  RET -
```

PARAMETRES -

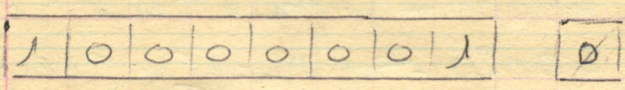
Puis MEMORY 39999
CALL 40000

DECALAGES

Déplacer bit par bit le contenu d'un registre vers la gauche ou la Droite. Le bit entrant est un Ø, le bit sortant va dans le Carry + Exception pour SRA où le bit n° 7 (bit signe) garde sa valeur (1 si bit n° 7 = 1 et non zéro entrant, les autres bits se décalent normalement → droite

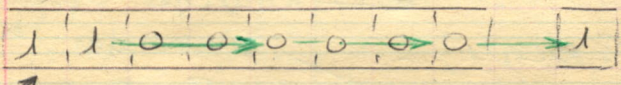
SRA reg. ou (HL) ou (IX + d) → droite

Avant SRA - = 129.



Carry = Ø -

Après SRA - = 192



Carry = 1 -

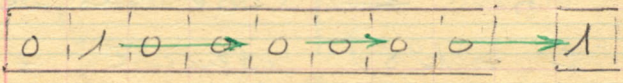
Le bit 7 garde sa valeur, et se décale. / Décalage Arithmétique

≈

SRL reg ou (HL) ou (IX + d) → droite :

avant = 129

après = 64.



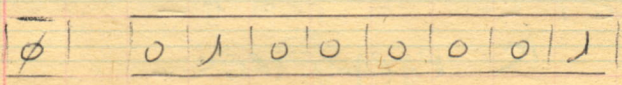
Carry = 1

bit entrant = Ø, bit expulsé dans Carry. / Décalage Logique -

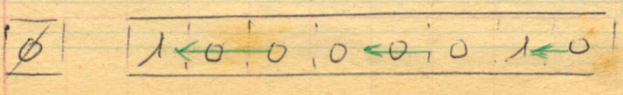
≈

SLA reg. ou (HL) ou (IX + d) ← Gauche

Avant = 65.



Après SLA = 130



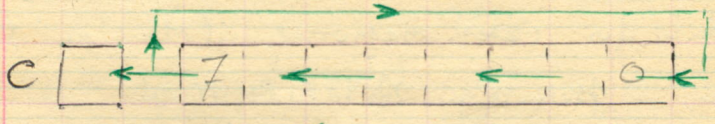
La lettre C est le CARRY ne participe pas -

ROTATIONS -

8 bits : Le bit expulsé va dans le Carry. (qui lui ne participe pas à la rotation) et se place en bit n° 7 ou n° 0

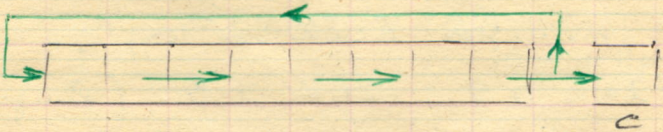
Rotations gauches (RL)	Rotations Droites (RR)
RLC A (ou RLCA)	RRC A (ou RRCA).
RLC Reg. / RLC (HL)	RRC Reg. / RRC (HL)
RLC (IX + d).	et RRC (IX + d) -

bit n° 7 devient 0 -



RLC

bit n° 0 devient n° 7 -



RRC -

9 bits : Le bit expulsé va dans le Carry, le Carry participe à la Rotation : Gauches

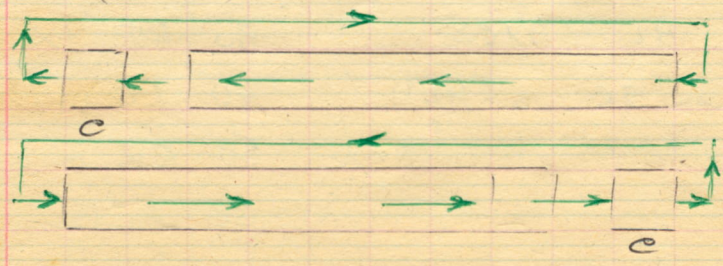
RLA RL Reg -

RL (HL) et RL (IX+d) -

Rotations - Droites -

RR A RR Reg -

RR (HL) RR (IX+d) -



RL

RR -

Reg = Registre simple - B, C, D, E, H, L,
(HL) et (IX + d) : sur adresse pointée par (HL) ou (IX + d)

Pour Tester Rotations et Décalages de A ou Reg -

```
10 ENT-
20 LD A, (Chiffre)
30 - (RLA ou RLCA etc).
40 CALL BB5A -
50 RET.
```

```
10 ENT-
20 LD Reg, (chiffre).
30 (RLA, RLCA etc)
40 CALL BB5A -
50 RET-
```

Pour des Décalages ou Rotation à travers les Registres pointeur sur des Cases mémoires (HL) ou (IX +d)
Annuler sur l'Assembleur AUTOFORMATION la Ligne 64018 puis un REM, modifier 6022 en remplaçant MODE1 par MODE2 : INK 0,9 : INK 1,3 - (Rouge sur Vert.)
Les déplacements sont visibles à l'écran sous la forme et l'emplacement des PIXELS.

Gauche/Droite - pour HL

```
10 ENT 49162
20 ORG 49162
30 LD HL, 49162
40 RL ou SR:__(HL)
50 RET.
```

idem pour IX+d -

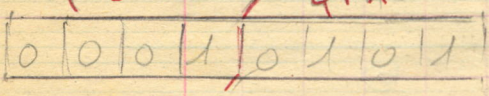
```
10 ENT 49152
20 ORG 49152
30 LD IX,49152
40 LD (IX+20) chiffre
50 RR ou ___ (IX+20)
60 RET
```

ROTATIONS 4 bits.

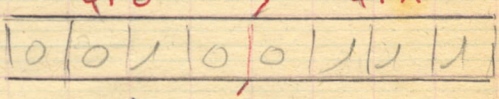
RLD - L'octet est séparé en 2 quartets ; QFO quartet fort et QFA Quartet FAible ! Le décalage s'effectue dans cet ordre a) - le QFA de HL va dans le QFO de HL / b) - le QFO de HL va dans le QFA de l'Accu / c) - le QFA de l'Accu va dans le QFA de HL.

AVANT.

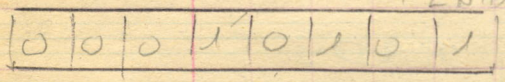
Accu = 37 — QFO | QFA



HL = H39 — QFO | QFA

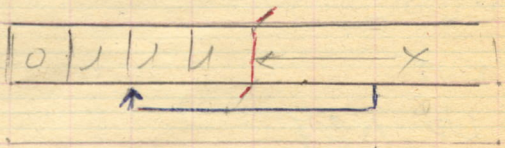


PENDANT - a).

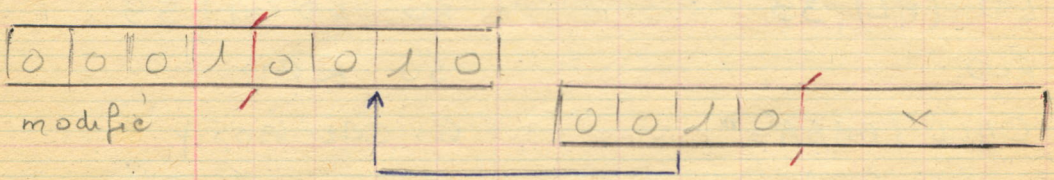


A inchangé.

QFA de HL ← QFO de HL.

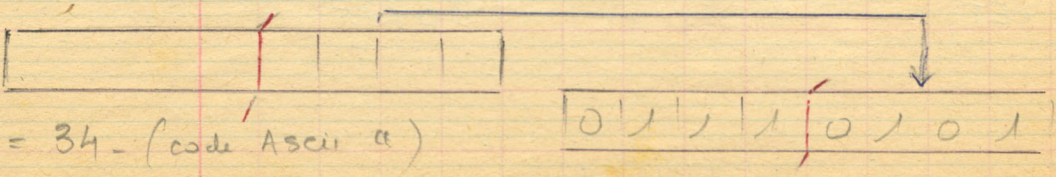


PENDANT b) QFO de HL dans QFA de l'Acc



A modifié

c) QFA de l'Acc → QFA de HL

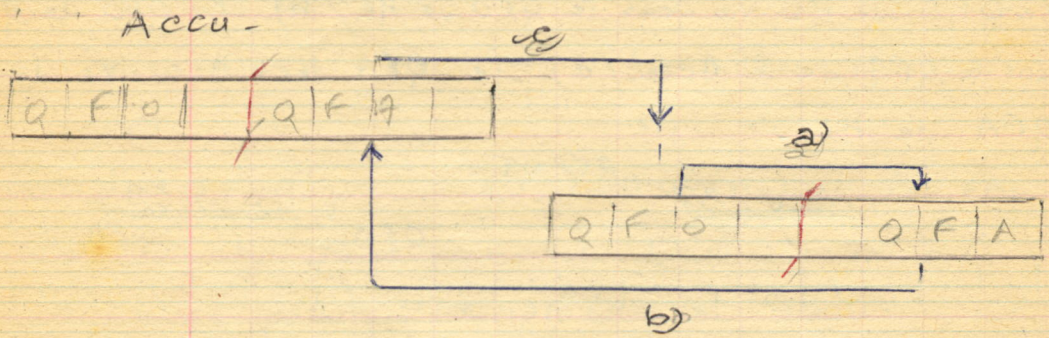


A = 34 - (code Ascii a)

HL = 117.
(code Ascii de a).

RRD : La rotation Droite s'effectue dans cet ordre :

a) QFO de HL → QFA de HL / b) QFA de HL dans QFA de l'Accu / c) QFA de l'Accu → QFO de HL



Listings

RLD :

```
10 ENT
20 LD A,37
30 LD HL,&A000
40 LD (HL),39
50 RLD
60 CALL &BB5A =" code Ascii = 34
70 LD A,(HL)   dans l'Accu
80 CALL &BB5A = u code Ascii = 117
90 RET         dans HL
```

RRD :

```
10 ENT
20 LD A,103
30 LD HL,&A000
40 LD (HL),39
50 RRD
60 CALL &BB5A = ' code Ascii =
70 LD A,(HL) 103 dans l'Accu
80 CALL &BB5A = r code Ascii
90 RET       114 dans HL
```

- PIRATAGE -

- Copie de disquette protégée -

avec RSexplogic par fichiers ; ou copie intégrale en écrasant les protections -

- Pour créer (ou Concer) les fichiers binaires pas évident -

- TRUCS ASTUCES -

CONT en mode direct relance le prg après ESC.
Les Virgules séparent la variable d'INPUT.
en lève le point d'interrogation en mode prg
Ex : 10 INPUT "Nombre=" ; a / 10 INPUT "Nombre=", a donne ? Nombre

pour extraire la √ carrée d'un nombre on ~~le multiplie~~ l'élève à la puissance 0.5 (Inverse de 2).
Ex : ~~2~~ √4 = (4 Exp 0.5) = ~~2~~ √9 = (9 ^ 0.5) = 3.
pour extraire la √ cubique d'un nombre on l'élève à la puissance 0.333 (Inverse de 3)
Ex : ∛8 = 8 ^ 0.333 ≈ 2 / ∛27 = 27 ^ 0,333 ≈ 3.
INT arrondit au 1er inférieur et enlève la partie fractionnaire -
Ex : 10.5463 devient avec INT (-10).
10 -
Pour conserver une partie fractionnaire avec INT. il faut multiplier le nombre 10.5463 par 100 ce qui fait passer 1054 dans la partie entière, puis le diviser par 100 ce qui devient 10.54 -
100 pour 2 décimales - car 100 = 10^2
En règle générale : pour imprimer un nbre X avec

D décimales - l'on utilise la formule :
PRINT INT(X * 10^D) / 10^D
D devient le nbre de déci - 10 = 1 déci / 100 = 2 / 1000 = 3
etc - —

TABLE D'ADDITION

Le signe de l'Addition est : +

1 et 1 font 2	4 et 1 font 5	7 et 1 font 8
1 — 2 — 3	4 — 2 — 6	7 — 2 — 9
1 — 3 — 4	4 — 3 — 7	7 — 3 — 10
1 — 4 — 5	4 — 4 — 8	7 — 4 — 11
1 — 5 — 6	4 — 5 — 9	7 — 5 — 12
1 — 6 — 7	4 — 6 — 10	7 — 6 — 13
1 — 7 — 8	4 — 7 — 11	7 — 7 — 14
1 — 8 — 9	4 — 8 — 12	7 — 8 — 15
1 — 9 — 10	4 — 9 — 13	7 — 9 — 16
1 — 10 — 11	4 — 10 — 14	7 — 10 — 17
2 et 1 font 3	5 et 1 font 6	8 et 1 font 9
2 — 2 — 4	5 — 2 — 7	8 — 2 — 10
2 — 3 — 5	5 — 3 — 8	8 — 3 — 11
2 — 4 — 6	5 — 4 — 9	8 — 4 — 12
2 — 5 — 7	5 — 5 — 10	8 — 5 — 13
2 — 6 — 8	5 — 6 — 11	8 — 6 — 14
2 — 7 — 9	5 — 7 — 12	8 — 7 — 15
2 — 8 — 10	5 — 8 — 13	8 — 8 — 16
2 — 9 — 11	5 — 9 — 14	8 — 9 — 17
2 — 10 — 12	5 — 10 — 15	8 — 10 — 18
3 et 1 font 4	6 et 1 font 7	9 et 1 font 10
3 — 2 — 5	6 — 2 — 8	9 — 2 — 11
3 — 3 — 6	6 — 3 — 9	9 — 3 — 12
3 — 4 — 7	6 — 4 — 10	9 — 4 — 13
3 — 5 — 8	6 — 5 — 11	9 — 5 — 14
3 — 6 — 9	6 — 6 — 12	9 — 6 — 15
3 — 7 — 10	6 — 7 — 13	9 — 7 — 16
3 — 8 — 11	6 — 8 — 14	9 — 8 — 17
3 — 9 — 12	6 — 9 — 15	9 — 9 — 18
3 — 10 — 13	6 — 10 — 16	9 — 10 — 19

TABLE DE MULTIPLICATION

Le signe de la Multiplication est : ×

1 fois 1 font 1	4 fois 1 font 4	7 fois 1 font 7
1 — 2 — 2	4 — 2 — 8	7 — 2 — 14
1 — 3 — 3	4 — 3 — 12	7 — 3 — 21
1 — 4 — 4	4 — 4 — 16	7 — 4 — 28
1 — 5 — 5	4 — 5 — 20	7 — 5 — 35
1 — 6 — 6	4 — 6 — 24	7 — 6 — 42
1 — 7 — 7	4 — 7 — 28	7 — 7 — 49
1 — 8 — 8	4 — 8 — 32	7 — 8 — 56
1 — 9 — 9	4 — 9 — 36	7 — 9 — 63
1 — 10 — 10	4 — 10 — 40	7 — 10 — 70
2 fois 1 font 2	5 fois 1 font 5	8 fois 1 font 8
2 — 2 — 4	5 — 2 — 10	8 — 2 — 16
2 — 3 — 6	5 — 3 — 15	8 — 3 — 24
2 — 4 — 8	5 — 4 — 20	8 — 4 — 32
2 — 5 — 10	5 — 5 — 25	8 — 5 — 40
2 — 6 — 12	5 — 6 — 30	8 — 6 — 48
2 — 7 — 14	5 — 7 — 35	8 — 7 — 56
2 — 8 — 16	5 — 8 — 40	8 — 8 — 64
2 — 9 — 18	5 — 9 — 45	8 — 9 — 72
2 — 10 — 20	5 — 10 — 50	8 — 10 — 80
3 fois 1 font 3	6 fois 1 font 6	9 fois 1 font 9
3 — 2 — 6	6 — 2 — 12	9 — 2 — 18
3 — 3 — 9	6 — 3 — 18	9 — 3 — 27
3 — 4 — 12	6 — 4 — 24	9 — 4 — 36
3 — 5 — 15	6 — 5 — 30	9 — 5 — 45
3 — 6 — 18	6 — 6 — 36	9 — 6 — 54
3 — 7 — 21	6 — 7 — 42	9 — 7 — 63
3 — 8 — 24	6 — 8 — 48	9 — 8 — 72
3 — 9 — 27	6 — 9 — 54	9 — 9 — 81
3 — 10 — 30	6 — 10 — 60	9 — 10 — 90

TABLE DE SOUSTRACTION

Le signe de la Soustraction est : —

1 de 2 reste 1	4 de 5 reste 1	7 de 8 reste 1
1 — 3 — 2	4 — 6 — 2	7 — 9 — 2
1 — 4 — 3	4 — 7 — 3	7 — 10 — 3
1 — 5 — 4	4 — 8 — 4	7 — 11 — 4
1 — 6 — 5	4 — 9 — 5	7 — 12 — 5
1 — 7 — 6	4 — 10 — 6	7 — 13 — 6
1 — 8 — 7	4 — 11 — 7	7 — 14 — 7
1 — 9 — 8	4 — 12 — 8	7 — 15 — 8
1 — 10 — 9	4 — 13 — 9	7 — 16 — 9
1 — 11 — 10	4 — 14 — 10	7 — 17 — 10
2 de 3 reste 1	5 de 6 reste 1	8 de 9 reste 1
2 — 4 — 2	5 — 7 — 2	8 — 10 — 2
2 — 5 — 3	5 — 8 — 3	8 — 11 — 3
2 — 6 — 4	5 — 9 — 4	8 — 12 — 4
2 — 7 — 5	5 — 10 — 5	8 — 13 — 5
2 — 8 — 6	5 — 11 — 6	8 — 14 — 6
2 — 9 — 7	5 — 12 — 7	8 — 15 — 7
2 — 10 — 8	5 — 13 — 8	8 — 16 — 8
2 — 11 — 9	5 — 14 — 9	8 — 17 — 9
2 — 12 — 10	5 — 15 — 10	8 — 18 — 10
3 de 4 reste 1	6 de 7 reste 1	9 de 10 reste 1
3 — 5 — 2	6 — 8 — 2	9 — 11 — 2
3 — 6 — 3	6 — 9 — 3	9 — 12 — 3
3 — 7 — 4	6 — 10 — 4	9 — 13 — 4
3 — 8 — 5	6 — 11 — 5	9 — 14 — 5
3 — 9 — 6	6 — 12 — 6	9 — 15 — 6
3 — 10 — 7	6 — 13 — 7	9 — 16 — 7
3 — 11 — 8	6 — 14 — 8	9 — 17 — 8
3 — 12 — 9	6 — 15 — 9	9 — 18 — 9
3 — 13 — 10	6 — 16 — 10	9 — 19 — 10

TABLE DE DIVISION

Le signe de la Division est : :

1 en 1 est 1 fois	4 en 4 est 1 fois	7 en 7 est 1 fois
1 — 2 — 2 —	4 — 8 — 2 —	7 — 14 — 2 —
1 — 3 — 3 —	4 — 12 — 3 —	7 — 21 — 3 —
1 — 4 — 4 —	4 — 16 — 4 —	7 — 28 — 4 —
1 — 5 — 5 —	4 — 20 — 5 —	7 — 35 — 5 —
1 — 6 — 6 —	4 — 24 — 6 —	7 — 42 — 6 —
1 — 7 — 7 —	4 — 28 — 7 —	7 — 49 — 7 —
1 — 8 — 8 —	4 — 32 — 8 —	7 — 56 — 8 —
1 — 9 — 9 —	4 — 36 — 9 —	7 — 63 — 9 —
1 — 10 — 10 —	4 — 40 — 10 —	7 — 70 — 10 —
2 en 2 est 1 fois	5 en 5 est 1 fois	8 en 8 est 1 fois
2 — 4 — 2 —	5 — 10 — 2 —	8 — 16 — 2 —
2 — 6 — 3 —	5 — 15 — 3 —	8 — 24 — 3 —
2 — 8 — 4 —	5 — 20 — 4 —	8 — 32 — 4 —
2 — 10 — 5 —	5 — 25 — 5 —	8 — 40 — 5 —
2 — 12 — 6 —	5 — 30 — 6 —	8 — 48 — 6 —
2 — 14 — 7 —	5 — 35 — 7 —	8 — 56 — 7 —
2 — 16 — 8 —	5 — 40 — 8 —	8 — 64 — 8 —
2 — 18 — 9 —	5 — 45 — 9 —	8 — 72 — 9 —
2 — 20 — 10 —	5 — 50 — 10 —	8 — 80 — 10 —
3 en 3 est 1 fois	6 en 6 est 1 fois	9 en 9 est 1 fois
3 — 6 — 2 —	6 — 12 — 2 —	9 — 18 — 2 —
3 — 9 — 3 —	6 — 18 — 3 —	9 — 27 — 3 —
3 — 12 — 4 —	6 — 24 — 4 —	9 — 36 — 4 —
3 — 15 — 5 —	6 — 30 — 5 —	9 — 45 — 5 —
3 — 18 — 6 —	6 — 36 — 6 —	9 — 54 — 6 —
3 — 21 — 7 —	6 — 42 — 7 —	9 — 63 — 7 —
3 — 24 — 8 —	6 — 48 — 8 —	9 — 72 — 8 —
3 — 27 — 9 —	6 — 54 — 9 —	9 — 81 — 9 —
3 — 30 — 10 —	6 — 60 — 10 —	9 — 90 — 10 —

FABRIQUE EN FRANCE