MEMENTO BASIC
TO7 / TO7-70

TO7 / TO7-70
BASIC MEMO

*RECOMMANDATIONS D'UTILISATION

Ne posez pas vos doigts sur la bande, ou sur les contacts de la cartouche, protégez votre logiciel après usage, évitez les températures élevées, l'humidité, la proximité des champs magnétiques...
Ne manipulez la cartouche que si l'unité centrale est à l'arrêt.
Le non respect de ces recommandations d'utilisation entraînera la déchéance immédiate de la garantie.

*GARANTIE

La garantie de TO TEK INTERNATIONAL couvre les défauts de fabrication des composants physiques du support de ce logiciel et les vices cachés.
Tout logiciel **renvoyé par votre revendeur TO TEK INTERNATIONAL** et reconnu défectueux après expertise, sera gratuitement échangé pendant 12 mois à compter de la date d'achat, le port étant à la charge de l'acheteur.
Attention : la garantie décrite ci-dessus n'est applicable qu'en FRANCE et pour un logiciel acheté sur le territoire français. Dans tous les autres cas, consultez votre revendeur TO TEK INTERNATIONAL local.

TO TEK International

MEMENTO
MEMO 7 BASIC TO7 / TO7-70

Pour commencer en BASIC, se reporter à "INITIATION AU BASIC TO7 / TO7-70"

Pour approfondir les instructions, consulter le "MANUEL DE RÉFÉRENCE BASIC TO7 / TO7-70"

Pour utiliser des disquettes, voir "LE BASIC D.O.S. DU TO7 / TO7-70 et du MO5"

MISE EN ROUTE DES ÉQUIPEMENTS

1 - Insérer la mémo7 BASIC dans le lecteur de mémo7,
2 - Mettre les périphériques sous tension,
3 - Mettre le téléviseur sous tension,
4 - Mettre l'ordinateur sous tension. Le menu apparaît sur l'écran.
5 - Régler le crayon optique (adapter la luminosité du téléviseur si nécessaire),
6 - Répondre au menu.

TOUCHES PARTICULIERES

STOP — Suspend l'exécution. Reprise par appui sur une autre touche.

CNT + C — Arrêt complet de l'exécution. Reprise par CONT

RAZ — Effacement de tout l'écran.

CNT + X — Effacement du reste de la ligne, à partir du curseur.

↑ ↓ ← → — Déplacements du curseur.

⮌ — Place le curseur en haut et à gauche de l'écran.

INS + EFF — Insertion d'un espace, effacement d'un caractère.

ENTREE — Validation d'une ligne.

• — Touche majuscule.

• + ⇧ ⇩ — Passage en mode majuscule ou minuscule (voir voyant *min)*

ACC — Touche accent. Doit être utilisée en mode minuscule et avant la lettre à accentuer. Appuyer sur la touche ACC, puis :

- pour les accents, simultanément sur

• + 7 — accent aigu

• + ø — accent grave

• + 2 — tréma

• + ù — accent circonflexe

- pour le c cédille, deux fois sur C

IMPRIMANTE PARALLÈLE

Liste du programme :

LIST "LPRT :" sur 40 colonnes
LIST "LPRT : (80)", 10-200 liste partielle sur 80 colonnes

Impression de données :

OPEN "O", # canal, "LPRT : (colonnes)"

PRINT # canal, USING "image" ; liste de données

CLOSE canal

FICHIERS SUR CASSETTE

Ecriture du fichier : Positionner la bande, appuyer sur la touche enregistrement
OPEN "O", # canal, "nom de fichier"

PRINT # canal, liste de données

CLOSE canal

Lecture du fichier : Positionner la bande et appuyer sur la touche lecture
OPEN "I", # canal, "nom de fichier"

INPUT # canal, liste de variables

CLOSE canal

FICHIERS GÉNÉRALISÉS

Nom de fichier : Jusqu'à 8 lettres ou chiffres, et un suffixe facultatif formé d'un point et trois lettres.

Descripteur de fichier :

Voie série	"COMM : (options) nom de fichier"
Cassette	"CASS : nom de fichier"
Imprimante parallèle	"LPRT : (nombre de colonnes)"
Ecran	"SCRN : (nombre de colonnes)"
Clavier	"KYBD :"

L'option de la voie série est formée de 3 à 5 chiffres :
Premier chiffre : vitesse

1	110 bauds	4	1200 bauds
2	300 bauds	5	2400 bauds
3	600 bauds	6	4800 bauds

Second chiffre : 7 ou 8 nombre de bits transmis par caractère
Chiffres suivants : 0 à 256 longueur de la ligne ou du bloc transmis

OPEN "O", # 3, "COMM : (48132)"

INSTRUCTIONS

Attention : voir l'encadré spécial TO7-70 à la fin de ce memento, notamment en ce qui concerne les instructions **SCREEN** et **COLOR,** et la fonction **POINT.**

Les exemples différents sont séparés par : . Les options sont en italiques.

ATTRB largeur, hauteur, *masquage*
Caractère standard = 0 double = 1
Caractère masqué = 1 non masqué = 0
ATTRB 1 , 1 , 1

AUTO *1re ligne, incrément*
Numérotation automatique des lignes
AUTO 5 , 7

BEEP Produit un son court

BOX et BOXF
Voir encadré relatif au graphique

CLEAR *nombre d'octets chaines, adresse max, nombre de caractères utilisateur*
Réservation de place pour travail sur les chaines et les caractères utilisateur. Plus haute adresse utilisable par le BASIC.
CLEAR 500 , 34000 : CLEAR , , 4

CLOSE *numéro de canal , . . .*
Fermeture de fichiers
CLOSE : CLOSE 2 , 3

CLS Effacement de l'écran

COLOR *couleur forme, couleur fond, inversion des couleurs*
Définit la couleur des caractères
COLOR 3 , 4 : COLOR , , 1

CONSOLE *1re ligne, dernière ligne, blocage des couleurs, défilement*
Définit une fenêtre de travail (1 = couleurs bloquées, 0 sinon) et son mode de défilement (0 = normal, 1 = lent, 2 = par page)
CONSOLE 10 , 15 : CONSOLE , , 1 , 1

CONT Poursuit l'exécution du programme interrompu par CNT C ou STOP

DATA donnée , . . .
Données du programme
DATA 3 , 4 , 5 , "BONJOUR"

DEFGR$(numéro) = n1 , n2 , . . ., n7 , n8
Définition d'un caractère utilisateur (numéro = 0 à 127 , n1 . . . n8 = 0 à 255). Le numéro maximum (+ 1) est spécifié par CLEAR , , 1
DEFGR$ (0) = 128 , 64 , 32 , 16 , 24 , 36 , 66 , 129

DEFINT plage , . . . variable entière
DEFSNG plage , . . . variable simple précision
DEFDBL plage , . . . variable double précision
DEFSTR plage , . . . variable de chaine
Définit le type des variables dont la première lettre est dans la plage indiquée
DEFDBL A-D , V-X

DEFUSR numéro = adresse
Adresse de début d'un sous-programme écrit en langage machine (numéro = 0 à 9)

DELETE plage
Supprime des lignes du programme
DELETE 100-150 : DELETE 100- : DELETE -150

DIM tableau (taille) , . . .
Dimensionne les tableaux
DIM A (10 , 15) , MOT$ (N)

END Fin de l'exécution

ERROR numéro d'erreur
Simule l'erreur numérotée

EXEC adresse
Exécution d'un sous-programme en langage machine à partir de l'adresse indiquée
EXEC 34500

FOR var = début TO fin *STEP incrément*
. . . Répétition des instructions entre FOR et NEXT jusqu'à ce que
. . . **var** atteigne la valeur *fin.* STEP précise l'incrément (1 s'il est omis)

NEXT *var , . . .*
FOR J = 10 TO 0 STEP - 1 : . . . . . : NEXT

GOSUB numéro de ligne
Appel du sous-programme Basic commençant à la ligne numérotée
GOSUB 100

GOTO numéro de ligne
Branche à la ligne numérotée
GOTO 250

IF condition THEN instructions 1 *ELSE instructions 2*
IF condition GOTO numéro de ligne *ELSE numéro de ligne*
Exécution des instructions 1 si la condition est réalisée et des instructions 2 sinon
IF A = 0 THEN PRINT "Solde nul" : GOTO 200 ELSE 100

INPEN Voir encadré "crayon optique"

INPUT *"message"*, var1 , . . .
Lecture de données introduites au clavier
Point-virgule après "message" affiche un point d'interrogation

INPUT # canal, variable, . . .
Lecture de données à partir d'un fichier séquentiel
INPUT # 1 , NOM$, PRENOM$, AGE , SALAIRE

INPUTPEN Voir encadré "crayon optique"

INPUTWAIT numéro de ligne, durée en secondes, *"message", liste de variables*
Lecture de données introduites au clavier dans un temps limite. Au delà, branchement à la ligne numérotée.
Point-virgule après "message" affiche un point d'interrogation
INPUTWAIT 1000 , 10 ; A, B

LET var = expression
Affectation. LET est facultatif
LET X1 = (-B-SQR (DELTA)) / (2 ★ A)

LINE Voir encadré relatif au graphique

LINEINPUT "message" ; variable chaine
LINEINPUT *"message"*, variable chaine
Lecture d'une ligne introduite au clavier
Point-virgule après "message" affiche un point d'interrogation
LINEINPUT PHRASE$

LINEINPUT # canal, variable chaine
Lecture d'une ligne de données à partir d'un fichier

LIST *plage*
Listage d'une partie ou de tout le programme sur l'écran
LIST : LIST 100-150 : LIST -80

LIST "descripteur de fichier", *plage*
Listage du programme sur l'imprimante ou vers un fichier

LOAD "descripteur de fichier", *R*
Charge en mémoire le programme indiqué. L'option R lance l'exécution

LOADM *"descripteur de fichier", décalage,* *R*
Charge en mémoire une page écran, ou un programme binaire avec translation en mémoire. L'option R lance l'exécution

LOCATE colonne, ligne, *mode*
Positionne le curseur sur l'écran (curseur invisible si mode = 0 et visible si mode = 1)
LOCATE 15 , 10

MERGE "descripteur de fichier", *R*
Fusionne le programme indiqué avec le programme présent en mémoire. L'option R lance l'exécution

MID$ (chaine 1 , i ,j) = chaine 2
j caractères de chaine 1 sont remplacés à partir du ième caractère, par les j premiers caractères de chaine 2
MID$ ("MACHINE-XXXXXXX" , 9 , 5) = "OUTIL" donne "MACHINE-OUTILXX"
Ne pas confondre avec la fonction MID$

MOTORON
Commmande le défilement de la cassette

MOTOROFF
Arrête le défilement de la cassette

NEW Détruit programme et variables en mémoire centrale

ON ERROR GOTO numéro de ligne
Branche au numéro de ligne indiqué dès qu'une erreur est détectée

ON variable GOTO ligne 1, *ligne 2, . . .*
ON variable GOSUB ligne 1, *ligne 2, . . .*
Branche au numéro de ligne indiqué selon la valeur de variable
ON T GOTO 100 , 150 , 200

ONPEN GOTO, ONPEN GOSUB
Voir encadré "crayon optique"

OPEN "mode", # canal, descripteur de fichier
Ouverture d'un canal en entrée (mode = I) ou en sortie (mode = O) pour un fichier ou un périphérique
OPEN "I" , # 1 , "Clients . ABC" : OPEN "O" , # 3 , "LPRT :"
Voir encadrés "imprimantes" et "fichiers"

PEN Voir encadré "crayon optique"

PLAY Voir encadré "son"

POKE adresse, i
Ecrit à l'adresse indiquée la valeur de i (i compris entre 0 et 255)
POKE 29952 , 23

PRINT expressions
Affichage de résultats ou de textes sur l'écran

PRINT # canal, expressions
Ecriture sur un fichier ou un périphérique identifié par son numéro de canal

PRINT USING image ; expressions
PRINT # canal USING image ; expressions
L'écriture se réfère à l'image spécifiée (chaine qui contient des descripteurs de mise en page)

Descripteurs pour chaines :

!	Premier caractère de la chaine uniquement
% %	n-2 espaces entre % permettent l'édition de n caractères
&	attribue une longueur suffisante pour l'édition de la chaine

Descripteurs pour valeurs numériques :

#	1 position numérique
.	sépare la partie entière de la partie décimale
+	édition du signe + si le nombre est positif
—	placé à droite : le signe moins est édité à droite de la valeur
★ ★	les espaces non significatifs sont remplacés par des ★
↑↑↑↑	utilisation de la notation avec exposant

Descripteur spécial :

=	le caractère suivant le signe = dans la chaine est édité tel quel

Tout autre caractère ou espace dans l'image est considéré comme séparateur et édité tel quel.
PRINT USING "###.## Francs" ; MONTANT

PSET Voir encadré "Instructions graphiques"

READ variable, . . .
Lecture des données des instructions DATA
READ NCLIENT, NOM$, PRENOM$

REM *commentaire* ou ' *commentaire*
Insertion d'un commentaire dans un programme

RESTORE *numéro de ligne*
Relecture des données au début de l'instruction DATA indiquée ou au début de la première instruction DATA si le numéro de ligne est omis

RESUME Sortie d'un sous programme de traitement d'erreur (appelé par ON ERROR GOTO) et rebranchement à l'instruction où l'erreur a été détectée

RESUME NEXT
Rebranche à l'instruction suivant celle qui a provoqué l'erreur

RESUME numéro de ligne
Rebranche au numéro de ligne indiqué

RETURN Sortie de sous-programme et retour au programme appelant

RUN *numéro de ligne*
Lancement de l'exécution à partir de la ligne indiquée (du début si le numéro est omis)

RUN *"descripteur de fichier", numéro de ligne*
Chargement du programme indiqué puis exécution
RUN : RUN "PROG1", 100

SAVE "descripteur de fichier", ***P ou A***
Sauvegarde d'un programme avec protection (P) ou en ASCII (A)
SAVE "SECRET" , P : SAVE "PROG1" , A

SAVEM "descripteur de fichier", adr. début, adr. fin, adr. lancement
Sauvegarde d'une page écran ou d'un programme écrit en langage machine
SAVEM "BINAIRE", 32700, 32750, 32702

SCREEN *couleur forme, couleur fond, couleur cadre, inversion couleurs*
Fixe les couleurs de l'écran
SCREEN 3 , 4 , 0 : SCREEN , , , 1
Ne pas confondre avec la fonction SCREEN

SCREENPRINT
Recopie de l'écran sur l'imprimante PR90-040

SKIPF *"nom"*
Saute, sur la cassette, le programme ou fichier indiqué, ou le premier rencontré si le nom est omis
SKIPF : SKIPF "PROG2"

STOP Suspend l'exécution du programme. On peut relancer par CONT

TRON Mise en route du mode TRACE

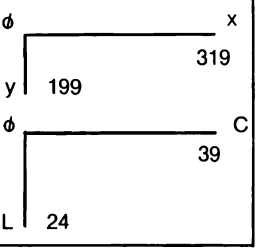
TROFF Arrêt du mode TRACE

UNMASK Démasque sur l'écran toutes les zones masquées par ATTRB

WAIT adresse, masque-et, ***masque-ou***
Suspend l'exécution en attente d'un événement. L'exécution reprend dès que :
masque-et AND (contenu de la mémoire pointée XOR masque-ou) est différent de zéro. Les opérations AND et XOR sont effectuées bit à bit sur les octets.

Couleurs	Forme (et fond en mode caractère)	Fond en mode graphique
noir	0	-1
rouge	1	-2
vert	2	-3
jaune	3	-4
bleu	4	-5
mauve (magenta)	5	-6
bleu clair (cyan)	6	-7
blanc	7	-8

coordonnées graphiques



OPÉRATEURS LOGIQUES

AND	ET logique	**NOT**	complément
EQV	equivalence	**OR**	OU logique (inclusif)
IMP	implication	**XOR**	OU exclusif

OPÉRATEURS ARITHMÉTIQUES

Le tableau suivant contient les opérateurs par priorité décroissante :

↑	élévation à la puissance	X↑ **Y**
—	moins unaire	**— X**
***./**	multiplication et division	**X * Y et X / Y**
ω	division entière	**X ω Y**
MOD	modulo	**X MOD Y**
+ . —	addition et soustration	**X + Y — Z**

Le caractère ↑ est obtenu en appuyant simultanément sur les touches [•] et ω̂

VARIABLES ERL et ERR

ERL et **ERR** sont des variables particulières qui permettent, lorsqu'une erreur est détectée pendant l'exécution d'un programme, de connaître le numéro de la ligne de l'erreur détectée et le numéro de l'erreur afin de la traiter.

INSTRUCTIONS GRAPHIQUES

BOX rectangle défini par 2 sommets opposés en diagonale
BOXF rectangle rempli
LINE segment défini par les coordonnées de ses extrémités
PSET point ou caractère

L'absence du couple (x1 , y1) ou (C1 , L1) provoque le tracé à partir de la position précédente

mode graphique

BOX *(x1 , y1)* - (x2 , y2), *couleur*
BOXF *(x1 , y1)* - (x2 , y2), *couleur*
LINE *(x1 , y1)* - (x2 , y2), *couleur*
PSET (x , y), *couleur*

mode caractère

Le tracé s'effectue au moyen de la première lettre de la chaine. BOXF remplit le rectangle avec cette lettre, PSET écrit la lettre au point (C , L).

BOX *(C1 , L1)* - (C2 , L2) chaine, *couleur forme, couleur fond, inversion de couleurs*

BOXF *(C1 , L1)* - (C2 , L2) chaine, *couleur forme, couleur fond, inversion de couleurs*

LINE *(C1 , L1)* - (C2 , L2) chaine, *couleur forme, couleur fond, inversion de couleurs*

PSET chaine, *couleur forme, couleur fond, inversion de couleurs*

BOX (39 , 24) - (20 , 13) "★" , 3 , 4
LINE (0 , 0) - (0 , 13) "ESSAI"
PSET (2 , 1) A $, 5

CRAYON OPTIQUE

PTRIG

Teste le contact du crayon optique.
Valeur "vraie" (-1) si crayon appuyé, "faux" (0) sinon
IF PTRIG GOTO 100

INPEN x , y

Lit les coordonnées du point visé par le crayon optique.
x = -1 et y = -1 si point visé en dehors de l'écran, ou mauvaise lecture

INPUTPEN x , y

Attend, pour lire, que le crayon soit appuyé

100 INPUTPEN X , Y : IF X <0 GOTO 100

PEN zone, *zone, . . .*

Définit des zones rectangulaires de lecture pour les ordres ONPEN.
Forme des zones : numéro ; (x1 , y1) - (x2 , y2)
("numéro" varie de 0 à 7)
PEN 0 ; (40 , 50) - (60 , 70) , 3 ; (40 , 110) - (60 , 130)
Si les coordonnées sont absentes, PEN supprime la zone
PEN 1 ; , 3 ;
PEN toutes les zones sont supprimées

ONPEN GOSUB ligne 0, *ligne 1, . . .*
ONPEN GOTO ligne 0, *ligne 1, . . .*

Attend que le crayon optique soit appuyé et branche à l'une des lignes, suivant le numéro de la zone visée

SON

PLAY mélodie, *mélodie, . . .*

Chaque mélodie est une chaine de caractères composée de notes :
DO ; RE ; MI ; FA ; SO ; LA ; SI ; P P = Pause
dont on peut modifier
la durée : L1 à L96 , (L24 au début de l'exécution)
l'octave : O1 à O5 , (O4 au début)
le tempo : T1 à T255 , (T5 au début)
l'attaque : A0 à A255 , (A0 au début)
Placer (éventuellement) après la note : dièse **#** ou bemol **b**

PLAY "DODODOREL48MIREL24DOMIREREL48DO"
A$ = "O3A5DOT1O2A1DOREMIFAO4A0T5SO # " : PLAY A$

Durées conventionnelles des notes

notes	standard	pointée	triolet
ronde	L96	-	L64
blanche	L48	L72	L32
noire	L24	L36	L16
croche	L12	L18	L8
double croche	L6	L9	L4
triple croche	L3	-	L2

FONCTIONS

ABS (v) Valeur absolue

ASC (chaine) Donne le code ASCII du premier caractère de la chaine

CDBL (v) Conversion en double précision

CHR$ (code) Donne le caractère dont le code ASCII est indiqué

CINT (v) Arrondit à l'entier le plus proche
CINT (13 . 9) donne 14

COS (v) Cosinus de l'angle exprimé en radians

CSNG (v) Conversion en simple précision

CSRLIN Donne le numéro de la ligne où se trouve le curseur
LOCATE0 , 5 : PRINT CSRLIN

DEFUSR i = adresse
Définit l'adresse du début du sous-programme i écrit en langage machine (i = 0 à 9)

EOF (canal) Donne la valeur vraie (-1) si une fin de fichier a été détectée, faux (0) sinon
IF EOF (1) THEN PRINT "Fin de fichier" : GOTO 900

EXP (V) Exponentielle

FIX (v) Troncature de v.

FRE (0) Donne la place totale libre en mémoire

FRE (A$) Donne la place libre pour le traitement des chaines

HEX$ (v) Conversion en hexadécimal

INKEY$ Prend au vol la dernière touche appuyée au clavier
T$ = INKEY$

INPUT$ (i) Lecture de i caractères au clavier sans affichage
A$ = INPUT$ (8)

INSTR (chaine, sous-chaine)
Donne la position de la sous-chaine dans la chaine

INSTR (i, chaine, sous-chaine)
La recherche commence à partir du i-ème caractère
INSTR (5 , "TELEVISEUR" , "E") donne 8

INT (v) Donne le plus grand entier inférieur ou égal à v
INT (13 . 9) donne 13

LEFT$ (chaine , i)
Prend les i premiers caractères de la chaine
LEFT$ ("MACHINE" , 3) donne "MAC"

LEN (chaine) Longueur de la chaine
LEN ("MACHINE") donne 7

LOG (v) Logarithme népérien

MID$ (chaîne , i , j)
Prend j caractères de la chaîne à partir du i-ème caractère
MID$ ("MACHINE" , 4 , 3) donne "HIN"
Ne pas confondre avec l'instruction MID$

OCT$ (v) Conversion en octal

PEEK (adresse)
Donne le contenu de l'octet dont l'adresse est indiquée

POINT (x , y) Donne par son numéro (entre -8 et + 7) la couleur du point de coordonnées x , y

POS *(canal)* Donne le numéro de la colonne à partir de laquelle sera effectuée l'impression (ou l'affichage si canal est omis)

PTRIG Donne -1 si le contact du crayon optique est fermé, 0 sinon

RIGHT$ (chaîne , i)
Donne les i derniers caractères de la chaîne
RIGHT$ ("MACHINE" , 5) donne "CHINE"

RND (u) Génération d'un nombre aléatoire dans l'intervalle 0 , -1 (le même nombre est redonné si u est négatif ou nul)

SCREEN (l , c) Donne le code ASCII du caractère situé sur l'écran aux coordonnées l , c
Ne pas confondre avec l'instruction SCREEN

SGN (v) Donne -1 si v est négatif, 0 si v est nul et 1 si v est positif

SIN (v) Sinus de l'angle v exprimé en radians

SPC (i) Utilisé avec PRINT : édition de i caractères "espace"
PRINT "A" ; SPC (2) ; "B"

SQR (v) Racine carrée de v

STICK (i) Donne la position (0 à 8) de la manette de jeu i (i = 0 ou 1)

STRIG (i) Donne -1 si le bouton de la manette de jeu i (i = 0 ou 1) est appuyé, 0 sinon

STR$ (v) Conversion d'une valeur numérique en une chaîne

TAB (c) Utilisé avec PRINT : provoque une tabulation à la colonne c

TAN (v) Tangente de l'angle v exprimé en radians

USR i, argument
Exécution avec passage de paramètres du sous-programme en langage machine numéro i (i = 0 à 9)

VAL (chaîne) Donne la valeur numérique du nombre représenté par les premiers caractères de chaîne ou 0 si chaîne ne représente pas un nombre

VARPTR (var) Donne l'adresse du premier octet de la variable.

ERREURS DÉTECTÉES

Code	numéro	
AO	52	Fichier déjà ouvert (Already Open)
BD	58	Mauvaises données dans le fichier (Bad Data)
BS	9	Indice en dehors des limites (Subscript out of range)
CN	17	Suite d'exécution impossible (Can't continue) - arrêt sur erreur - modification du programme après une interruption par CNT - C
DD	10	Définition multiple ou définition d'un tableau après l'avoir utilisé avec un dimensionnement implicite (redimensioned array)
DS	56	Commande directe dans un fichier ASCII en cours de chargement (Direct Statement)
DU	60	Périphérique non disponible (Device unavailable)
FC	5	Argument incorrect dans l'appel d'une fonction ou de certaines instructions (Illegal function call)
FD	55	Descripteur de fichier incorrect (Bad file descriptor)
FM	51	Mode d'accès au fichier incorrect (Bad file mode)
FN	23	Instruction FOR sans instruction NEXT (For without Next)
ID	12	Instruction interdite en mode immédiat (Illegal Direct)
IE	54	Tentative de lecture d'un fichier après la fin de fichier Pour éviter cette erreur, utiliser la fonction EOF (Input past End)
IO	53	Erreur d'entrée/sortie (Input/Output)
IU	59	Périphérique en cours de fonctionnement (Device in use)
LS	15	Chaîne de caractères trop longue (String too long)
MO	22	Opérande manquant (Missing Operand)
NF	1	Instruction NEXT sans instruction FOR ou NEXT avec variable ne correspondant pas à l'instruction FOR exécutée (Next without For)
NO	57	Fichier non ouvert (Not Open)
NR	19	Instruction RESUME manquante dans le sous-programme de traitement des erreurs (No Resume)
NU	50	Numéro de fichier incorrect (Not in use)
OM	7	Dépassement de capacité en mémoire centrale (Out of Memory)
OD	4	Instruction READ alors que les données des instructions DATA sont épuisées (Out of Data)
OS	14	Manque de place pour les chaînes (Out of string Space)
OV	6	Dépassement de capacité par valeur supérieure (Overflow)
PP	61	Programme protégé (Protected program)
RE	20	Instruction RESUME alors qu'il n'y a pas eu d'erreur détectée (Resume without Error)
RG	3	Instruction RETURN alors que GOSUB n'a pas été exécuté (Return without Gosub)
SN	2	Erreur de syntaxe (Syntax error)
ST	16	Expression sur chaîne trop complexe (String formula Too complex)
TM	13	Non correspondance de type (Type Mismatch)
UE	21	Erreur non définie (Undefined Error)
UF	18	Fonction utilisateur non définie (Undefined user Fonction)
UL	8	Numéro de ligne inexistant (Undefined Line)
/0	11	Division par zéro

PARTICULARITÉS DU TO7-70
Capacité mémoire et gestion des 8 couleurs supplémentaires

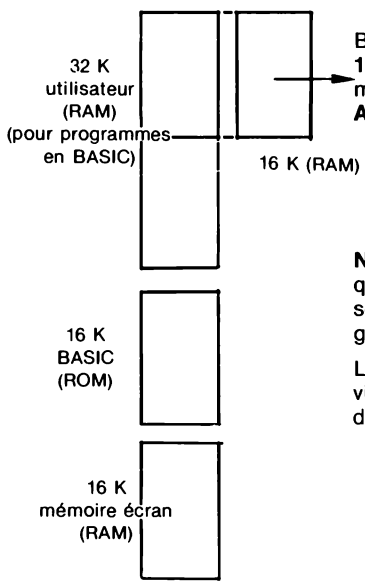
Le TO7-70 se différencie du TO7 essentiellement par sa **capacité mémoire, qui a été accrue,** et par l'emploi possible de **16 couleurs** au lieu de 8...

I. Exploitation de la mémoire du TO7-70 avec le BASIC Version 1.0 MICROSOFT

Le TO7 a 8 K de mémoire-utilisateur.

Dans sa version de base, le TO7-70 dispose, lui, de **48 K de mémoire vive utilisateur, dont seuls 32 K sont accessibles simultanément.** Un **bloc-mémoire (ou banque-mémoire) de 16 K est** utilisable pour vos programmes grâce à la **technique de la commutation de blocs-mémoire,** même si cette technique n'est pas applicable directement au moyen d'instructions BASIC : il vous faut des **sous-programmes spéciaux en Assembleur, incorporables dans vos programmes BASIC.**

STRUCTURE DE LA MÉMOIRE DU TO7-70



Bloc (ou banque) supplémentaire **de 16 K (RAM), utilisable par** recouvrement (overlay) pour vos programmes en **ASSEMBLEUR et par commutation.**

N.B. : Le mot "recouvrement" veut dire que ces 16 K sont aux mêmes adresses physiques que le bloc de 16 K de gauche.

Les 16 K de la cartouche BASIC ne viennent jamais masquer une partie des 32 K utilisateurs.

A titre d'**exemples,** voici deux programmes utilisant le bloc-mémoire (ou banque-mémoire) non accessible directement par le BASIC :

1. Un programme pour recopier une disquette lorsqu'on ne dispose que d'un seul lecteur de disquettes ;
2. Un programme pour mémoriser et rappeler des écrans graphiques générés de façon aléatoire ; l'exemple utilise les 8 premières couleurs...

```
10 PRINT" Programme de backup disque rapide avec un seul lecteur"
15 PRINT" Assurez vous que la disquette copie est initialisee (sinon appuyez sur
 CNT C.  inserez la disquette copie.Puis commande DSKINI 0. Puis relancez ce Pro
gramme   par RUN)"
20 CLEAR,&H9FDF
30 GOSUB 1000 'Initialisation routine de commutation
40 A$="Inserez la disquette source,Puis frappez une touche quelconque"
50 B$="Inserez la disquette copie.Puis frappez une touche quelconque"
60 C$=""
70 X=VARPTR(C$)
80 POKE X,128
100 FOR I=0 TO 2
110 IO=1
120 MAX=15
130 PRINTA$
140 R$=INPUT$(1)
150 PT=&HA000 A=USR0(0) 'Selection banque 0
160 IF I=2 THEN MAX=7
170 FOR P=0 TO MAX
180 IF P=8 THEN PT=&HA000 A=USR0(1) 'Selection banque 1
190 FOR S=1 TO 16
200 POKE X+1,INT(PT/256)
210 POKE X+2,(PT-&HA000) MOD 256
215 PRINTP+I*16,S,HEX$(PT)
220 ON IO GOSUB 500,600
230 PT=PT+128
240 NEXT S,P
250 IF IO=1 THEN IO=2 PRINTB$ GOTO 140
260 NEXT I
270 PRINT"Copie terminee
280 END
500 MID$(C$,1)=DSKI$ 0,P+I*16,S)
510 RETURN
600 DSKO$ 0,P+I*16,S,C$
610 RETURN
1000 'Programme de commutation banque
1010 ' Le numero de banque est donne de 0 a 5
1020 DATA 52,86,206,231,192,230,75,196,251,231,75,166,3,48,140,10,166,134,167,73
 ,202,4,231,75,53,214,15,23,231,103,167,39
1030 FOR I=&H9FE0 TO &H9FFF
1040 READ A POKE I,A
1050 NEXT
1060 DEFUSR0=&H9FE0
1070 RETURN
```

```
10 CLEAR,&H9F9F
20 GOSUB 1000 'Initialisation routine de commutation
30 GOSUB 2000 'Initialisation routine de copie ecran
40 SCREEN,0,0
50 LOCATE 0,0,0
100 'Dessin aleatoire
110 CLS
120 FOR I=1 TO 100
130 LINE(160,100)-(RND*320,RND*200),INT(RND*8)
140 NEXT
150 A=USR0(0)'Selection 1ere banque
160 A=USR1(0)'Copie ecran --> memoire
200 '2eme dessin aleatoire
210 CLS
220 FOR I=1 TO 50
230 BOX(RND*320,RND*200)-(RND*320,RND*200),INT(RND*8)
240 NEXT
250 A=USR0(1) 'Selection 2eme banque
260 A=USR1(0) 'Copie ecran --> memoire
300 ' recuperation des images
310 CLS
320 PRINT"Quelle image (0 ou 1) ? "
330 R%=VAL(INPUT$(1))
340 IF R%<0 OR R%>1 GOTO 320
350 A=USR0(R%) 'Selection bonne banque
360 A=USR1(1) 'Copie memoire --> ecran
370 R$=INPUT$(1) 'On attend un caractere pour continuer
380 GOTO 310
1000 'Programme de commutation banque
1010 ' Le numero de banque est donne de 0 a 5
1020 DATA 52,86,206,231,192,230,75,196,251,231,75,166,3,48,140,10,166,134,167,73
,202,4,231,75,53,214,15,23,231,103,167,39
1030 FOR I=&H9FE0 TO &H9FFF
1040 READ A POKE I,A
1050 NEXT
1060 DEFUSR0=&H9FE0
1070 RETURN
2000 'Programme de copie
2010 ' de l'ecran vers la memoire si 0 est donne
2020 ' de la memoire vers l'ecran si 1 est donne
2030 DATA 52,118,166,3,142,64,0,206,160,0,77,39,2,30,19,182,231,195,138,1,183,23
1,195,16,142,16,0,236,129,237,193,49,63,38,248,182,231,195,68,36,16,72,140,96,0
39,5,206,64,0,32,224,142,64,0,32,219,53,246
2040 FOR I=&H9FA0 TO &H9FDA
2050 READ A POKE I,A
2060 NEXT
2070 DEFUSR1=&H9FA0
2080 RETURN
```

Vous y trouverez **la routine de commutation de blocs-mémoire dans un programme BASIC, qui est la suivante :**

N° Ligne		Commentaire :
	CLEAR,&H9FDF	Réserve de la place pour le sous-programme ASSEMBLEUR dans la zone des 16 K-octets fixe du BASIC.
...	GOSUB XXXX	Initialise le sous-programme.
...	A = USR0(n)	Sélectionne le bloc-mémoire n. n = 0 ou 1 pour le TO7-70. n = 0,1,2,3,4,5 avec l'extension mémoire 64 K-octets. A la mise en service, le bloc 0 est sélectionné.
XXXX	DATA 52, ...	Introduction du code machine.
	FOR I = &H9FE0 TO &H9FFF READ A : POKE I, A	Lecture du code machine (READ) et introduction en mémoire (POKE)
...	NEXT	
...	DEFUSR0 = &H9FE0	
...	RETURN	

Cette routine est utilisable dans tous vos programmes en BASIC...

Remarque au sujet de la capacité mémoire et de la fonction FRE (0) qui donne la place disponible — en octets — dans la mémoire centrale : puisqu'il y a toujours un bloc-mémoire de 16 K RAM présent et un autre qui est "masqué", PRINT FRE (0) ne fera jamais apparaître les 48 K de mémoire vive utilisateur du TO7-70.

Mais grâce à la technique de commutation des blocs, le bloc masqué peut être rendu "visible" et celui qui est visible prendre sa place à tout moment...

II. Exploitation des 8 couleurs supplémentaires du TO7-70

Le TO7-70 dispose de 16 couleurs fond et forme pour les caractères.

ATTENTION : En BASIC, les instructions **SCREEN** (pour changer les couleurs de la fenêtre de travail et de l'entourage de l'écran) et **COLOR** (pour changer les règles d'utilisation des couleurs de l'écran pour les prochains caractères qui seront affichés) n'acceptent que des arguments inférieurs à 8.

Vous pouvez accéder aux 8 couleurs supplémentaires avec les ordres suivants :

Soit :

F = 112 : couleur caractère ou forme
F = 120 : couleur fond
F = 128 : couleur tour

C = 0 : gris
C = 1 : rose
C = 2 : vert clair
C = 3 : jaune poussin
C = 4 : bleu ciel
C = 5 : rose parme
C = 6 : cyan clair
C = 7 : orange

L'instruction **SCREEN** pour le fond et la forme se fera par :

```
PRINT CHR$(27)CHR$(32)CHR$(F + C)
```

Exemple :

Si vous voulez des caractères vert clair sur un fond rose parme, vous devrez **successivement** rentrer les instructions suivantes, soit directement, soit dans un programme :

— Pour les caractères, F = 112
— Pour vert clair, C = 2
— donc F + C = 114

Instruction :

```
PRINT CHR$(27)CHR$(32)CHR$(114)
```

— Pour le fond, F = 120
— Pour rose parme, C = 5
— donc F + C = 125

Intruction :

```
PRINT CHR$(27)CHR$(32)CHR$(125)
```

Pour le pourtour de l'écran (F = 128), il suffit de faire :

```
PRINT CHR$(27)CHR$(F + C)
```

Même chose pour l'instruction **COLOR.**

Quant à la fonction **POINT** (qui donne une valeur numérique indiquant la couleur du point de l'écran de coordonnées X et Y), elle retournera des valeurs comprises entre — 16 et 15 selon les couleurs :

FOND	FORME	
— 16	15	orange
— 15	14	cyan clair
— 14	13	rose parme
— 13	12	bleu ciel
— 12	11	jaune poussin
— 11	10	vert clair
— 10	9	rose
— 9	8	gris

Illustration :

Voici maintenant un programme de défilement des 16 couleurs disponibles dans le TO7-70 avec la cartouche BASIC version 1.0 :

```
10  CLS
20  FOR I = 0 TO 7
30  FOR J = 0 TO 10
40  SCREEN, I
50  NEXT J
60  NEXT I
70  FOR I = 8 TO 15
80  FOR J = 0 TO 5
90  PRINT CHR$(27)CHR$(32)CHR$(112 + I)
100 NEXT J
110 NEXT I
120 GOTO 20
```

Commentaires :

10	Efface l'écran
20.60	Définit le compteur de boucle pour l'affichage des 8 premières couleurs (0 à 7) accessibles par SCREEN.
30.50	Définit le compteur de temporisation de la couleur sur l'écran.
40	Transforme le fond dans la nouvelle couleur.
70.110	Définit le compteur de boucle pour l'affichage des 8 couleurs supplémentaires non accessibles par SCREEN.
80.110	Définit le compteur de temporisation de la couleur sur l'écran.
90	Transforme le fond dans la nouvelle couleur 120 + (I — 8) = 112 + I.
120	Recommence la séquence.
	Terminer le programme par l'appui simultané de CNT et C.

JEU DE CODES ASCII

Code décimal		Code décimal		Code décimal		Code décimal		Code décimal		Code décimal	
00	NUL	22	SS2 Touche caracteres	42	*	64	@	85	U	107	k
01			accentués ACC	43	+	65	A	86	V	108	l
02	STOP touche STOP clavier	23		44	,	66	B	87	W	109	m
03		24	CAN Efface la fin de la ligne	45	—	67	C	88	X	110	n
04		25		46	.	68	D	89	Y	111	o
05		26		47	/	69	E	90	Z	112	p
06		27	ESC Appel d'une séquence	48	0	70	F	91	(	113	q
07	SONNETTE	28	INS (INS clavier)	49	1	71	G	92		114	r
08	BS (Retour arrière) ← clavier	29	DEL (EFF clavier)	50	2	72	H	93	)	115	s
09	HT (Tabulation H) → clavier	30	RS Touche clavier ←	51	3	73	I	94	^	116	t
10	LF (Saut de ligne) ↓ clavier	31	US Séparateur d'article	52	4	74	J	95		117	u
11	VT (Tabulation V) ↑ clavier	32	Espace	53	5	75	K	96		118	v
12	FF (Saut de page) RAZ clavier	33	!	54	6	76	L	97	a	119	w
13	CR (Retour à la ligne) ENTREE	34	"	55	7	77	M	98	b	120	x
14	SO Mode semi-graphique	35	#	56	8	78	N	99	c	121	y
15	SI Mode alphanumérique	36	$	57	9	79	O	100	d	122	z
16		37	%	58	:	80	P	101	e	123	{
17	DC1 Clignotement curseur	38	&	59	;	81	Q	102	f	124	
18	DC2 Répétition	39	'	60	<	82	R	103	g	125	}
19		40	(	61	=	83	S	104	h	126	
20	DC4 Arrêt curseur	41	)	62	>	84	T	105	i	127	■
21				63	?			106	j		

CODES ECRAN SPECIAUX

TOUCHE ACC + CARACTERE

Code décimal	Caractère	Résultat	Code décimal	Caractère	Résultat
35	#	£	122	z	œ
49	1	±	44	,	→
60	<	1/4	45	—	↑
61	=	1/2	46	.	←
62	>	3/4	47	/	↓
56	8	⁻¹⁄₋	106	j	OE

TO7 / TO7-70 BASIC MEMO

To start in BASIC see "INITIATION TO BASIC TO7 / TO7-70"
For further instructions see "TO7 / TO7-70 BASIC REFERENCE MANUAL"
To use the diskettes see "TO7 / TO7-70 / MO5 BASIC D.O.S."

GETTING STARTED

1 - Push the BASIC cartridge into the slot provided for it,
2 - Switch on the peripherals,
3 - Switch on the screen,
4 - Switch on the computer. The menu will be displayed on the TV screen.
5 - Regulate the light pen (adjust the brightness of the screen if necessary),
6 - Respond to the menu.

SPECIAL KEYS

STOP — Stops execution. To continue the program press any other key.

CTRL + C — Brings execution to a complete halt. Restart with CONT.

CLS — Clears the whole screen.

CTRL + X — Erases the end of the line from where the cursor points.

↑ ↓ ← → — Moves the cursor.

⇦ — Moves the cursor to the top left hand corner of the screen.

INS DEL — Inserts a blank space. Deletes a character.

ENTER — Line validation.

• — Upper case key.

• + ⇧ or ⇩ — Move to upper or lower case mode (see min. light)

ACC — Accent key. To be used in lower case and before the letter to be accented.
Press the ACC key, and then
- for accents, simultaneously on

• + é — acute accent
• + è ø — grave accent
• + ¨ 2 — dieresis
• + â ω — circumflex accent

- for the cedilla, press twice on c

PARALLEL PRINTER

Program list :

```
LIST "LPRT :"                  over 40 columns
LIST "LPRT : (80)", 10-200     partial list over 80 columns
```

Printing data :

```
OPEN "O", # channel, "LPRT : (columns)"
-------
PRINT # channel, USING "image" ; data list
-------
CLOSE channel
```

FILES ON CASSETTE

Writing a file : Put the tape in place, press the recording key

```
OPEN "O", # channel, "file name"
-------
PRINT # channel, data list
-------
CLOSE channel
```

Reading a file : Put the tape in place and press the play key

```
OPEN "I", # channel, 'file name"
-------
INPUT # channel, variable list
-------
CLOSE channel
```

FILES IN GENERAL

File name : Up to 8 letters or digits and an optional suffix made up of a period and three letters.

File descriptor :

Serial channel	"COMM : (options) file name"
Cassette	"CASS : file name"
Parallel printer	"LPRT : (number of columns)"
Screen	"SCRN : (number of columns)"
Keyboard	"KYBD :"

The option of the serial channel is made up of 3 to 5 digits:
First digit : speed

1	110 bauds	4	1200 bauds
2	300 bauds	5	2400 bauds
3	600 bauds	6	4800 bauds

Second digit : 7 or 8 number of bits transmitted per character
Following digits : 0 to 256 length of the line or of the block transmitted

OPEN "O", # 3, "COMM : (48132)"

INSTRUCTIONS

Refer to the TO7-70 box at the end of this document. Please note changes to instructions **SCREEN** and **COLOR** and Function **POINT.**

The different examples are separated by : . Options are in italics.

ATTRB width, height, *masking*
Standard character = **0** double = 1
Masked character = 1 unmasked = **0**
ATTRB 1 , 1 , 1

AUTO *1st line, increment*
Automatic line numbering
AUTO 5 , 7

BEEP Produces a short beeping sound

BOX and BOXF
See paragraph relevant to graphics

CLEAR *number of bytes for strings, maximum address, number of user characters.*
Reserves space for strings and user-defined characters
Highest address used by BASIC.
CLEAR 5**0**0 , 34**0**0**0** : CLEAR , , 4

CLOSE *channel number , . . .*
Close files
CLOSE : CLOSE 2 , 3

CLS Clears the screen

colour *Form colour, background colour, colour inversion*
Defines the colour of characters
COLOR 3 , 4 : COLOR , , 1

CONSOLE *1st line, last line, colour blocking, scroll speed*
Defines a working window (1 = colours blocked, **0** if not) and its scroll speed (**0** = normal, 1 = slow, 2 = by page)
CONSOLE 1**0** , 15 : CONSOLE , , 1 , 1

CONT Resumes the execution of a program interrupted by CTRL C or STOP

DATA data , . . .
Program data
DATA 3 , 4 , 5 , "HELLO"

DEFGR$(number) = n1 , n2 , . . . n8
Defines a user graphic character (number = **0** to 127 , n1 . . . n8 = **0** to 255). Maximum number is specified by CLEAR , , 1
DEFGR$ (**0**) = 128 , 64 , 32 , 16 , 24 , 36 , 66 , 129

DEFINT range , . . . integer variable
DEFSNG range , . . . simple precision variable
DEFDBL range , . . . double precision variable
DEFSTR range , . . . string variable
Defines the variable type whose first letter is within the indicated range
DEFDBL A-D , V-X

DEFUSR number = address
Address of the beginning of a subroutine written in machine language (number = 0 to 9)

DELETE range
Deletes the lines of the program
DELETE 100-150 : DELETE 100- : DELETE -150

DIM table (size) , . . .
Gives dimensions of tables
DIM A (10 , 15) , MOT$ (N)

END Ends the excecution of a program.

ERROR error number
Simulates a numbered error

EXEC address
Executes a subroutine written in machine language starting from the indicated address
EXEC 34500

FOR var = beginning TO end *STEP increment*
. . . Repeats the instructions between FOR and NEXT until **var** reaches
. . . the value **end.** STEP defines the increment (1 if it is omitted).

NEXT *var* , . . .
FOR J = 10 TO 0 STEP - 1 : . . . . . : NEXT

GOSUB line number
Branches to a BASIC subroutine starting from the numbered line
GOSUB 100

GOTO line number
Branches to the numbered line
GOTO 250

IF condition THEN instructions 1 *ELSE instructions 2*
IF condition GOTO line number *ELSE line number*
Executes the instructions 1 if the condition is filled and instructions 2 if not
IF A = 0 THEN PRINT "Balance null" : GOTO 200 ELSE 100

INPEN See section on "light pen"

INPUT *"message",* var1 , . . .
Reads data entered from the keyboard
Semi-colon after "message" displays question mark.

INPUT # channnel, variable, . . .
Reads data from sequential access file.
INPUT # 1 , NAME$, FIRST NAME$, AGE , SALARY

INPUTPEN See section on "light pen"

INPUTWAIT line number, duration in seconds, *"message", list of variables*
Reads data from the keyboard in a limited time. Beyond, branches to the numbered line.
Semi-colon after "message" displays a question mark
INPUTWAIT 1000 , 10 ; A, B

LET var = expression
Assigns a value to a variable. LET is optional.
LET X1 = (-B-SQR (DELTA)) / (2 ★ A)

LINE See section relevant to graphics

LINEINPUT *"message"*; character string
Enters a line from the keyboard
Semi-colon after "message" displays a question mark
LINEINPUT PHRASE$

LINEINPUT # channel, character string
Reads a line of data from a file

LIST *range*
Listing of part or of whole program on the screen
LIST : LIST 100-150 : LIST -80

LOAD "file descriptor", *R*
Loads an indicated program into memory Option R starts the execution

LOADM *"file descriptor", shift,* *R*
Loads the screen image into memory, or a binary program with an offset into memory. Option R starts the execution

LOCATE column, line, *mode*
Positions the cursor on the screen (cursor invisible if mode = 0 and visible if mode = 1)
LOCATE 15 , 10

MERGE "file descriptor", *R*
Merges the indicated program with the program in memory. Option R starts the execution

MID$ (string 1 , i ,j) = string 2
j characters of string 1 are replaced, starting from character i, by the first j characters of string 2
MID$ ("MACHINE-XXXXXXX" , 9 , 4) = "TOOL". gives "MACHINE-TOOLXX"
Not to be confused with the MID$ function

MOTORON
Starts the program recorder magnetic tape

MOTOROFF
Stops the tape of the program recorder

NEW Clears the program and destroys all the variables in memory

ON ERROR GOTO line number
Branches to a specified line number if an error is detected

ON variable GOTO line 1, *line 2, . . .*
ON variable GOSUB ligne 1, *line 2, . . .*
Branches to a specified line number according to the value of the variable ON T GOTO 100 , 150 , 200

ONPEN GOTO, ONPEN GOSUB
See section on "light pen"

OPEN "mode", # channel, file descriptor
Opens an input channel (mode = I) or an output channel (mode = O) for a file or a peripheral
OPEN "I" , # 1 , "Customers . ABC" : OPEN "O" , # 3 , "LPRT :"
See section on "printers" and "files"

PEN See section on "light pen"

PLAY See section on "sound"

POKE address, i
Places a byte with a value of i at the given address i (i between 0 and 255)
POKE 29952 , 23

PRINT expressions
Displays results or texts on the screen

PRINT # channel, expressions
Writes on a file or a peripheral identified by its channel number

PRINT USING image ; expressions
PRINT # channel USING image ; expressions
Printouts are subjected to the specified image (a string containing layout indicators)

Indicators for character strings :
! First character in the string only
% % n-2 spaces between % allows printing of n characters
& assigns sufficient for a string printout

Indicators for digital expressions :
represents the position of each decimal figure
. separates the integer part from the decimal part
+ Prints the sign + if the value is positive
— on the right side : the minus sign is printed on the right of the value
★ ★ non-significant spaces are replaced by ★
↑↑↑↑ exponential notation is being used

Special indicator :
= the character following the = sign in the string is printed as it is

Any other character or space on the screen is considered as a separator and printed as it is.
PRINT USING "###.##" ; AMOUNT

PSET See section on "graphic instructions"

READ variable, . . .
Reads the data of the instructions DATA
READ NCUSTOMER, NAME$, FIRST NAME$

REM *comment* or ' *comment*
Inserts a comment into a program

RESTORE *line number*
Rereads the data at the beginning of the specified DATA instruction or at the beginning of the first DATA instruction if the line number is omitted

RESUME Leaves an error subroutine (called by ON ERROR GOTO) and resumes at the instruction that triggered the error

RESUME NEXT
Resumes execution at the instruction following the one which triggered the error

RESUME line number
Resumes at the line number indicated

RETURN Leaves the subroutine and returns to the main program

RUN *line number*
Executes a resident program starting from the specified line (at the beginning if there is no number)

RUN *"file descriptor", line number*
Loads the specified program and executes it
RUN : RUN "PROG1", 100

SAVE "file descriptor", P or A
Saves a program with protection (P) or in ASCII (A)
SAVE "SECRET", P : SAVE "PROG1", A

SAVEM "file indicator", addr. beginning, addr. end, addr. execution
Saves a screen image or a binary program
SAVEM "BINARY", 32700, 32750, 32702

SCREEN *character colour, background colour, frame colour, colour inversion.*
Fixes the colours on the screen
SCREEN 3 , 4 , 0 : SCREEN , , , 8
Not to be confused with SCREEN function

SCREENPRINT
Copies the content of the screen onto the printer PR90-040

SKIPF *"name"*
Skips the first program or file on tape or the first found if the name is omitted
SKIPF : SKIPF "PROG2"

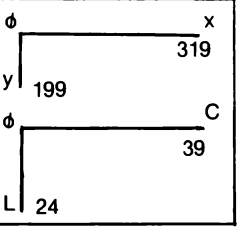
STOP Suspends execution of the program.
Execution can be resumed using CONT

TRON Displays the line numbers of instructions actually executed (Trace)

TROFF Cancels TRON

UNMASK Unmasks all the zones of the screen masked by ATTRB

WAIT address, and-mask, *or-mask*
Stops execution and waits for an outside signal. The execution of the program restarts when :
and-mask AND (contents of the memory XOR or-mask) is different from zero. The operations AND and XOR are executed bit by bit on the bytes.

Colours	Form (and backround in character mode)	Background in graphic mode	Graphic coordinates
black	0	-1	
red	1	-2	
green	2	-3	
yellow	3	-4	
blue	4	-5	
violet	5	-6	
light blue	6	-7	
white	7	-8	

LOGICAL OPERATORS

AND	Logical AND	**NOT**	Complement
EQV	Equivalence	**OR**	Logical OR (inclusive)
IMP	Implication	**XOR**	Exclusive OR

ARITHMETICAL OPERATORS

The table below shows the operators in descending priority order :

↑	To the power of	X↑ **Y**
—	Unary minus	**— X**
***./**	Multiplication and division	**X * Y and X / Y**
∖	Integer division	**X ∖ Y**
MOD	Modulo check	**X MOD Y**
+ . —	Add and subtract	**X + Y — Z**

The character ^ is obtained by pressing keys [•] and ∖ simultaneously.

ERL and ERR VARIABLES

ERL and **ERR** are specific variables that allow error line and error number to be displayed when an error is detected during program execution. This allows error processing and correction.

GRAPHIC INSTRUCTIONS

BOX A rectangle defined by 2 diagonally opposite vertexes
BOXF A rectangle filled in
LINE A segment defined by the coordinates of its extremities
PSET Point or character

If (x1 , y1) or (C1 , L1) is omitted, the segment is drawn from the previous position

graphic mode

BOX *(x1 , y1)* - (x2 , y2), *colour*
BOXF *(x1 , y1)* - (x2 , y2), *colour*
LINE *(x1 , y1)* - (x2 , y2), *colour*
PSET (x , y), *colour*

character mode

The plot is made from the first letter of the string. BOXF fills the rectangle with this letter, PSET writes the letter at the point (C , L).

BOX *(C1 , L1)* - (C2 , L2) string, *form colour, background colour, colour inversion*

LINE *(C1 , L1)* - (C2 , L2) string, *form colour, background colour, colour inversion*

PSET string, *form colours, background colour, colour inversion*

BOX (39 , 24) - (20 , 13) "★" , 3 , 4
LINE (0 , 0) - (0 , 13) "TEST"
PSET (2 , 1) A $, 5

LIGHT PEN

PTRIG

Checks the contact of the light pen.
Gives the value "true" (-1) if in contact, and "false" (0) if not
IF PTRIG GOTO 100

INPEN x , y

Reads the coordinates of the point indicated by the light pen. x = -1 and y = -1 if the point is off the screen or if the read is bad

INPUTPEN x , y

Reads only if the light pen is in contact with the screen

100 INPUTPEN X , Y : IF X <0 GOTO 100

PEN zone, *zone, . . .*

Defines rectangular reading zones for ONPEN instructions.
Form of zones : number ; (x1 , y1) - (x2 , y2)
("number" ranges from 0 to 7)
PEN 0 ; (40 , 50) - (60 , 70) , 3 ; (40 , 110) - (60 , 130)
If there are no coordinates, PEN cancels the zone
PEN 1 ; , 3 ;
PEN all zones are cancelled

ONPEN GOSUB line 0, *line 1, . . .*
ONPEN GOTO line 0, *line 1, . . .*

Waits until the light pen is in contact and branches to one of the lines according to the number of the chosen zone.

SOUND

PLAY melody, *melody, . . .*

Each melody is a character string composed of notes :
DO ; RE ; MI ; FA ; SO ; LA ; SI ; P P = Pause
where the following can be modified :
duration : L1 to L96 , (L24 at the beginning of execution)
octave : O1 to O5 , (O4 at beginning)
tempo : T1 to T255 , (T5 at beginning)
attack : A0 to A255 , (A0 at beginning)
A sharp # or a flat b can be placed after the note

```
PLAY "DODODOREL48MIREL24DOMIREREL48DO"
A$ = "O3A5DOT1O2A1DOREMIFAO4A0T5SO#" : PLAY A$
```

Standard note lengths

notes	standard	dotted	triplet
breve	L96	-	L64
minim	L48	L72	L32
crotchet	L24	L36	L16
quaver	L12	L18	L8
semiquaver	L6	L9	L4
demisemiquaver	L3	-	L2

FUNCTIONS

ABS (v) Absolute value

ASC (string) Gives the ASCII code of the first character in the string

CDBL (v) Converts into a double precision

CHR$ (code) Gives the character whose ASCII code value is indicated

CINT (v) Rounds off to the nearest integer. CINT (13 . 9) gives 14

COS (v) Calculates the cosine of an angle expressed in radians

CNSG (v) Converts into real "simple precision"

CSRLIN Gives the line number where the cursor is located
LOCATE0 , 5 : PRINT CSRLIN

DEFUSR i = address
Defines the address of the beginning of the subroutine i written in machine language (i = 0 to 9)

EOF (channel) Gives the value of (-1) if the end of a sequential file is reached, (0) if not
IF EOF (1) THEN PRINT "End of file" : GOTO 900

EXP (V) Calculates the exponential

FIX (v) Truncates v

FRE (0) Gives the remaining available space in memory

FRE (A$) Gives the remaining available space for processing character strings

HEX$ (v) Conversion into hexadecimal

INKEY$ Returns the last character string typed on the keyboard
T$ = INKEY$

INPUT$ (i) Reads i characters from the keyboard without display
A$ = INPUT$ (8)

INSTR (string, substring)
Gives the position of the substring in the string

INSTR (i, string, substring)
The search begins from the ith character
INSTR (5 , "TELEVISION" , "S") gives 7

INT (v) Gives the greatest integer smaller or equal to v
INT (13 . 9) gives 13

LEFT$ (string , i)
Takes the first i characters in the string
LEFT$ ("MACHINE" , 3) gives "MAC"

LEN (string) Gives the length of the character string
LEN ("MACHINE") gives 7

LOG (v) Napier logarithm

MID$ (string , i , j)
Takes j characters of the string starting from the ith character
MID$ ("MACHINE" , 4 , 3) gives "HIN"
Not to be confused with the instruction MID$

OCT$ (v) Converts value to a base of 8

PEEK (address)
Indicates the content of the byte at a given address

POINT (x , y) Gives a digital value between -8 and 7 indicating the colour of the point with the coordinates x , y

POS *(channel)* gives the column number from which printing will start (or display if the channel is omitted)

PTRIG Gives -1 if the light pen is in contact, 0 if not

RIGHT$ (string , i)
Gives the i last characters of a string
RIGHT$ ("MACHINE" , 5) gives "CHINE"

RND (u) Produces a real pseudo-random number in the interval 0 , -1 (the same number is given if u is less than or equal to 0)

SCREEN (l , c) Gives the ASCII code of the character displayed at position l , c on the screen

SGN (v) Gives -1 if v is negative, 0 if v is 0, and 1 if v is positive

SIN (v) Sine whose argument is expressed in radians

SPC (i) Used with PRINT only : prints i "space" characters
PRINT "A" ; SPC (2) ; "B"

SQR (v) Calculates the square root of v

STICK (i) Gives the position (0 to 8) of the joystick i(i = 0 or 1)

STRIG (i) Indicates the state of the joystick. -1 if the contact button is pressed, 0 if not

STR$ (v) Converts the value of a numeric expression into a string

TAB (c) Used with PRINT only ; places the cursor at column c

TAN (v) Tangent whose argument is expressed in radians

USR I, argument
Execution of subroutine i written in machine language with transfer of parameters (i = 0 to 9)

VAL (string) Gives the numerical value of the number represented by the first characters of the string or 0 if the string does not represent a number

VARPTR (var) Defines the address of the first byte of the variable.

ERROR MESSAGES

Code	number	
AO	52	File Already Open
BD	58	Bad Data
BS	9	Subscript out of range
CN	17	Can't continue - halt on error - program modified after interruption by CNT - C
DD	10	Redimensioned array (multiple definition of a table after being used with implicit dimensioning)
DS	56	Direct Statement (Direct Statement in ASCII file being loaded in)
DU	60	Device unavailable (Peripheral unavailable)
FC	5	Illegal function call (argument incorrect in calling up the function or certain instructions)
FD	55	Bad file descriptor
FM	51	Bad file mode
FN	23	FOR without NEXT (FOR instruction without NEXT instruction)
ID	12	Illegal direct (illegal instruction in immediate mode)
IE	54	Input past END (Attempt to read a file after end of file to avoid this error use the EOF function)
IO	53	Input / Output (Error)
IU	59	Device in use (peripheral in use)
LS	15	String too long (character string too long)
MO	22	Missing Operand
NF	1	NEXT without FOR (NEXT instruction without FOR instruction or NEXT with variable not corresponding to FOR instruction executed)
NO	57	Not Open (file not open)
NR	19	No RESUME (RESUME instruction missing in error processing sub-routine)
NU	58	Not in Use (File numberr incorrect)
OM	7	Out of Memory (Overflow of central memory capacity)
OD	4	Out of Data (READ instruction while data in DATA instructions is exhausted)
OS	14	Out of String Space (No space for strings)
OV	6	Overflow (overflow of capacity due to higher value)
PP	61	Protected program
RE	20	RESUME without error (RESUME instruction although no errors occured)
RG	3	RETURN without GOSUB (RETURN instruction although GOSUB has not been executed)
SN	2	Syntax error
ST	16	String formula too complex
TM	13	Type Mismatch
UE	21	Undefined Error (see page 51)
UF	18	Undefined user Function
UL	8	Undefined Line (Line number nonexistent)
/0	11	Division by zero

TO7-70 FEATURES
Increased memory capacity and management of eight extra colors

The main difference between the TO7-70 and the original TO7, is the **increased memory capacity** and the availability of **16 colors** instead of eight.

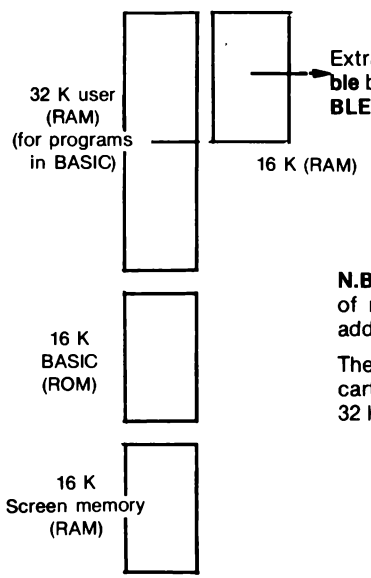
II. Using the TO7-70 memory with MICROSOFT BASIC Version 1.0

The TO7 has 8 K of user memory.

In its basic version TO7-70 has **48 K of user read/write memory. Only 32 K of memory is simultaneously accessible.**

An extra 16 K memory-block (or memory-bank) can be used for your programs thanks to the **memory-block switching routine.** This feature is not directly accessible using BASIC instructions. **Specific Assembly language sub-programs that can be included in your BASIC programs are required.**

TO7-70 MEMORY STRUCTURE



N.B. : "Overlay" means that the 16 K of memory is at the same physical address as the left 16 K block.

The 16 K space taken by the BASIC cartridge never masks a part of the 32 K user memory.

As an example here are two programs that use the memory-bock (or memory bank) that is not directly accessible by BASIC :

1. A program to copy a diskette when only one disk drive is provided ;
2. A program to store and recall graphic displays generated randomly. The example uses the first eight colors.

```
10 CLEAR,&H9F9F
20 GOSUB 1000 ' Switching routine initialisation
30 GOSUB 2000 ' Screen copy routine initialisation
40 SCREEN,0,0
50 LOCATE 0,0,0
100 ' Random drawing
110 CLS
120 FOR I=1 TO 100
130 LINE(160,100)-(RND*320,RND*200),INT(RND*8)
140 NEXT
150 A=USR0(0)'Select 1st bank
160 A=USR1(0)'Screen copy memory
200 ' 2nd random drawing
210 CLS
220 FOR I=1 TO 50
230 BOX(RND*320,RND*200)-(RND*320,RND*200),INT(RND*8)
240 NEXT
250 A=USR0(1) ' Select 2nd bank
260 A=USR1(0) ' Screen copy          memory
300 ' Drawing retreival
310 CLS
320 PRINT" "Which drawing (0 or 1)?"
330 R%=VAL(INPUT$(1))
340 IF R%<0 OR R%>1 GOTO 320
350 A=USR0(R%) Select correct bank
360 A=USR1(1)Memory copy          screen
370 R$=INPUT$(1) A character is needed to continue
380 GOTO 310
1000 'Bank switching program
1010 ' Bank number is given from 0 to 5
1020 DATA 52,86,206,231,192,230,75,196,251,231,75,166,3,48,140,10,166,134,167,73
,202,4,231,75,53,214,15,23,231,103,167,39
1030 FOR I=&H9FE0 TO &H9FFF
1040 READ A POKE I,A
1050 NEXT
1060 DEFUSR0=&H9FE0
1070 RETURN
2000 'Copy program
2010 ' from screen to memory if 0 is given
2020 ' from memory to screen if 1 is given
2030 DATA 52,118,166,3,142,64,0,206,160,0,77,39,2,30,19,182,231,195,138,1,183,23
1,195,16,142,16,0,236,129,237,193,49,63,38,248,182,231,195,68,36,16,72,140,96,0,
39,5,206,64,0,32,224,142,64,0,32,219,53,246
2040 FOR I=&H9FA0 TO &H9FDA
2050 READ A POKE I,A
2060 NEXT
2070 DEFUSR1=&H9FA0
2080 RETURN
```

```
10 PRINT" Fast disk back up program with one disk drive"
15 PRINT" Check that the copy disk is initialized (if not press
 ENT C insert the copy diskette. Then enter command DSKINI 0, then
restart the program using RUN)"
20 CLEAR,&H9FDF
30 GOSUB 1000 '  Switching routine initialisation
40 A$=" Insert source diskette then press any key"
50 B$=" Insert copy diskette then press any key"
60 C$=""
70 X=VARPTR(C$)
80 POKE X,128
100 FOR I=0 TO 2
110 IO=1
120 MAX=15
130 PRINTA$
140 R$=INPUT$(1)
150 PT=&HA000 A=USR0(0) Select bank 0
160 IF I=2 THEN MAX=7
170 FOR P=0 TO MAX
180 IF P=8 THEN PT=&HA000 A=USR0(1) Select bank 1
190 FOR S=1 TO 16
200 POKE X+1,INT(PT/256)
210 POKE X+2,(PT-&HA000) MOD 256
215 PRINTF+I*16,S,HEX$(PT)
220 ON IO GOSUB 500,600
230 PT=PT+128
240 NEXT S,P
250 IF IO=1 THEN IO=2 PRINTB$ GOTO 140
260 NEXT I
270 PRINT" Copy finished
280 END
500 MID$(C$,1)=DSKI$(0,P+I*16,S)
510 RETURN
600 DSKO$ 0,P+I*16,S,C$
610 RETURN
1000 'Bank switching program
1010 ' Bank number is given from 0 to 5.
1020 DATA 52,86,206,231,192,230,75,196,251,231,75,166,3,48,140,10,166,134,167,73
,202,4,231,75,53,214,15,23,231,103,167,39
1030 FOR I=&H9FE0 TO &H9FFF
1040 READ A POKE I,A
1050 NEXT
1060 DEFUSR0=&H9FE0
1070 RETURN
```

The **memory-block switching routine present in a BASIC program** is given here :

Line number		Comments :
	CLEAR,&H9FDF	Reserves space for the ASSEMBLER sub-program in the fixed BASIC 16 k-bytes area.
	GOSUB XXXX	Initializes the sub-program.
	A = USR0(n)	Selects memory-block n. n = 0 or 1 for TO7-70. n = 0,1,2,3,4,5 with the 64 K-byte memory extension. On initialisation block 0 is selected.
.		
.		
.		
.		
XXXX	DATA 52, . . .	Enter machine code.
	FOR I = &H9FE0 TO &H9FFF READ A : POKE I, A	Read machine code (READ) and enter into memory (POKE).
. . .	NEXT	
. . .	DEFUSR0 = &H9FE0	
. . .	RETURN	

This routine can be used in all of your BASIC programs.

Note on memory capacity and on function FRE (0) that gives the memory space in bytes, available in the main memory :
As there is always a 16 K RAM memory block and another that is "masked", PRINT FRE (0) will never show the 48 K TO-70 user read-write memory.

Using the block switching feature, the masked block can be made "usable" and the visible memory can take its place at any time.

II. Using the eight extra TO7-70 colors

The TO7-70 has 16 character and background colors.

NOTE : In BASIC instructions **SCREEN** (to change the color of the window and screen surrounding) and **COLOR** (to change the screen color use rules for the characters to be displayed) will only accept a number less than 8.

The extra eight colors can be accessed using these orders :

F = 112 : character or shape color
F = 120 : background color
F = 128 : surround color

with :

C = 0 : grey
C = 1 : pink
C = 2 : light green
C = 3 : yellow
C = 4 : light blue
C = 5 : dark pink
C = 6 : light cyan
C = 7 : orange

The **SCREEN** instruction for background and shape is entered by :

```
PRINT CHR$(27)CHR$(32)CHR$(F + C)
```

Example :

To obtain light green characters on a dark pink background the following instructions must be entered **successively** either directly or in a program :

— For characters, F = 112
— For light green, C = 2
— Giving F + C = 114

Instruction :

```
PRINT CHR$(27)CHR$(32)CHR$(114)
```

— For background, F = 120
— For dark pink, C = 5
— Giving F + C = 125

Intruction :

```
PRINT CHR$(27)CHR$(32)CHR$(125)
```

For the screen surround (F = 128) enter :

```
PRINT CHR$(27)CHR$(F + C)
```

And the same for instruction **COLOR.**

For the **POINT** function (that gives a numerical value for the color of the screen point identified by coordinates X and Y) values from —16 to 15 will be given depending on the colors :

BACKGROUND	SHAPE	
— 16	15	orange
— 15	14	light cyan
— 14	13	dark pink
— 13	12	light blue
— 12	11	yellow
— 11	10	light green
— 10	9	pink
— 9	8	grey

Operation :

Here is a program that shows the 16 colors available with TO7-70 and version 1.1. BASIC cartridge :

```
10  CLS
20  FOR I = 0 TO 7
30  FOR J = 0 TO 10
40  SCREEN, I
50  NEXT J
60  NEXT I
70  FOR I = 8 TO 15
80  FOR J = 0 TO 5
90  PRINT CHR$(27)CHR$(32)CHR$(112 + 1)
100 NEXT J
110 NEXT I
120 GOTO 20
```

Comments :

10	Clears the screen
20.60	Defines the loop counter for the display of the first eight colors (0 to 7) accessible by SCREEN.
30.50	Defines the delay counter for the color on the screen.
40	Changes the background ot the new color.
70.110	Defines the loop counter for the display of the extra eight colors not accessible by SCREEN.
80.110	Defines the screen color delay counter.
90	Changes the background to the new color 112 = 120 + (I—8).
120	Restarts the sequence.
	End program by pressing both CTRL and C simultaneously.

SET OF ASCII CODES

Decimal code		Decimal code		Decimal code		Decimal code		Decimal code		Decimal code	
00	NUL	22	SS2 key Accented character	42	*	64	a	85	U	107	k
01				43	+	65	A	86	V	108	l
02	STOP keyboard STOP button	23		44	.	66	B	87	W	109	m
03		24	CAN end of line erasure	45	—	67	C	88	X	110	n
04		25		46	.	68	D	89	Y	111	o
05		26		47	/	69	E	90	Z	112	p
06		27	ESC calls a sequence	48	0	70	F	91	(	113	q
07	BELL or BEEP	28	INS (INS keyboard)	49	1	71	G	92	'	114	R
08	BS (Backspace) ← Keyboard	29	DEL (EFF Keyboard)	50	2	72	H	93	)	115	s
09	HT (H tabulation) → Keyboard	30	RS Keyboard ↑	51	3	73	I	94	^	116	t
10	LF (Line Feed) ↓ Keyboard	31	US Unit separator	52	4	74	J	95	[ ]	117	u
11	VT (Vertical Tabulation) ↑ Keyboard	32	Space	53	5	75	K	96	£	118	v
12	FF (Form feed) - Keyboard CLS	33	!	54	6	76	L	97	a	119	w
13	CR (Carriage return) ENTER	34	"	55	7	77	M	98	b	120	x
14	SO Semi-graphic mode	35	#	56	8	78	N	99	c	121	y
15	SI Alphanumeric mode	36	$	57	9	79	O	100	d	122	z
16		37	%	58	:	80	P	101	e	123	{
17	DC1 Cursor Flashing	38	&	59	;	81	Q	102	f	124	|
18	DC2 Repitition	39	'	60	<	82	R	103	g	125	}
19		40	(	61	=	83	S	104	h	126	[ ]
20	DC4 Cursor off	41	)	62	>	84	T	105	i	127	■
21				63	?			106	j		

SPECIAL SCREEN CODES

TOUCHE ACC + CARACTERE

Decimal code	Character	Result	Decimal code	Character	Result
35	#	£	122	z	œ
49	1	±	44	.	→
60	<	1/4	45	—	↑
61	=	1/2	46	,	←
62	>	3/4	47	/	↓
56	8	:	106	j	OE

*USER ADVICE

Do not touch the tape, or the cartridge contacts, take good care of your programs after use, avoid high temperatures, humidity, and magnetic fields.
Do not insert or remove the cartridge unless the computer is switched off.
If this advice is not respected, the guarantee is void.

*GUARANTEE

The TO TEK INTERNATIONAL guarantee covers manufacturing defects od hardware components supporting the software, and hidden defects.
All software returned by your TO TEK INTERNATIONAL dealer and acknowledged after examination to be defective ; will be replaced free of charge during a period of 12 months from the date of purchase. Transportation costs are the responsibility of the purchaser.
NOTE : The above guarantee is valid only in FRANCE and for software purchased on FRENCH territory. Otherwise, consult your local TO TEK INTERNATIONAL dealer.

TO TEK International
B.P. 112
93175 Bagnolet Cedex
France

Imprimerie C I F C - 781 55 50