Réalisez votre ordinateur individuel

Les claviers la carte IVG 09 mode d'emploi du basic sur cassette

COMME l'annoncent les sous titres de cet article, le contenu promet d'être assez chargé, aussi allons-nous sans plus tarder entrer dans le vif du sujet.

Le clavier

Nous vous avions presque promis une solution le mois dernier ; c'est chose faite aujourd'hui puisque nous sommes à même de vous proposer deux claviers parfaitement adaptés à notre réalisation, aisément disponibles pour longtemps et, qui plus est, à un prix compatible avec un budget d'amateur.

Pourquoi deux claviers ? Pour répondre à tous les besoins ; en effet, le premier clavier proposé convient à la majorité des réalisateurs « moyens » (toute idée péjorative mise à part) tandis que le second permet de réaliser un ensemble de haut niveau ; nous allons voir pourquoi avec leur présentation ci-après.

Les deux claviers sont de marque et de fabrication anglaise et sont distribués en France par FACIM qui en dispose d'ores et déjà en stock et qui pourra vous renseigner sur les prix exacts de ceux-ci. Ils sont tous deux livrés complets avec leur électronique disposant de deux sorties : une sortie 8 bits ASCII parallèle directement compatible de nos cartes IVG ancien système et IVG09 dont nous allons parler ci-après, et une sortie série asynchrone qui n'est pas utilisée dans cette application. Ils utilisent une alimentation monotension 5 V, ce qui facilite leur mise en œuvre.

Côté mécanique, les contacts sont quasiment inusables puisqu'... inexistants ; en effet, comme dans les réalisations de haut de gamme, nous avons choisi des claviers capacitifs, c'est-à-dire que les touches n'actionnent aucun contact mais se contentent de déplacer un piston métallique en regard du circuit imprimé qui les supporte. L'on comprend alors facilement qu'un tel clavier soit quasiment inusable puisqu'il n'y a aucun contact mobile, et que les seules pièces en mouvement sont le piston constitué par la touche et son ressort de rappel ; de plus, ces éléments sont aisément interchangeables.

L'allure générale de ces deux claviers est excellente et, si les photos ne permettent pas bien d'en juger, il faut savoir que les touches ne sont pas montées directement sur le circuit imprimé mais sur un châssis en métal très rigide, peint en noir, sous lequel est fixé le circuit imprimé. Ce circuit imprimé est en verre époxy double face à trous métallisés et est d'excellente qualité. La fixation de ces claviers dans un boîtier est grandement facilitée par cette conception mécanique puisqu'il suffit de s'occuper de la fixation du châssis métallique pour que l'ensemble soit monté de façon rigide. De plus, ce qui n'est pas à négliger, la découpe à réaliser dans la face supérieure du boîtier pour laisser passer les touches est relativement simple car celles-ci sont disposées de façon aussi géométrique que possible.

Ces généralités seront complètes lorsque vous saurez que ces claviers consomment 200 mA sous 5 V, que leur signal de strobe est négatif et fait 10 μs de large, ce qui les rend directement compatibles, non seulement de IVG et IVG09 mais aussi du terminal vidéo de décembre 1981, et que les signaux sont disponibles sur un connecteur à 2 fois 8 contacts pour câble plat ; connecteur également approvisionné par FACIM.

Le clavier 63 touches

Ce clavier est celui que nous conseillons à l'utilisateur « moyen », c'est-à-dire à ceux d'entre vous qui vont faire avec leur mini-ordinateur de la programmation, un peu de texte (courrier par exemple) et, éventuellement, un peu de gestion. En effet, ce clavier dispose des touches classiques ASCII, des majuscules et des minuscules, ainsi que d'une touche ALPHA LOCK permettant le verrouillage des touches « lettres » en majuscules. Quatre touches de déplacement de curseur sont également à la disposition de l'utilisateur ainsi qu'une touche DELETE qui permet d'effacer la ligne en cours, et ce, sur tous nos logiciels. Toutes les touches disposent par ailleurs d'une possibilité de répétition automatique lorsque l'on y maintient le doigt appuyé. Enfin, la touche TAB, qui n'est d'aucune utilité dans nos logiciels, est programmée pour commander un effacement d'écran avec retour du curseur en haut et à gauche.

La figure 1 vous donne les cotes mécaniques exactes de ce clavier pour l'intégrer dans le boîtier de votre choix. Nous vous informons à ce sujet que TEKO et Schroff proposent des boîtiers bien adaptés à ce genre de clavier pour un prix de l'ordre de 200 F ; il ne reste plus qu'à faire la découpe dans la plaque frontale ! La figure 2 quant à elle vous indique le brochage du connecteur à 2 × 8 contacts qui véhicule tous les signaux. Pour le raccordement à la carte IVG, à la carte IVG09 ou au terminal vidéo de décembre 1981, reportez-vous à la figure 5 qui donne la correspondance entre les signaux issus de ce clavier et les appellations employées dans les articles relatifs à ces diverses cartes. La liaison entre le clavier et les cartes, quelles qu'elles soient, sera réalisée en câble ou en câblage conventionnel, la longueur de celle-ci pouvant atteindre deux mètres sans problème.

Pour en finir avec ce clavier 63 touches, sachez qu'il est référencé AKL 81-042 et qu'il est proposé à moins de 1 000 F.

Le clavier 117 touches

Ce clavier est plutôt destiné à ceux d'entre vous qui envisagent une utilisation intensive de leur mini-ordinateur ou qui pensent faire de la gestion et du traitement de texte avec celui-ci. En effet, ce clavier se compose d'une partie centrale à 63 touches analogues au clavier précédent. A gauche de cet ensemble se trouve un bloc de 15 touches comprenant les commandes de déplacement du curseur ainsi que des touches aux fonctions particulières suivant les logiciels mis en œuvre dans l'ordinateur. A droite de ce bloc central se trouve un clavier numérique disposant de plus des touches · , +, – et ENTER. Enfin, au-dessus de ces trois ensembles, une rangée de 23 touches de fonctions complète le tout. Ces touches seront utilisées pour activer directement

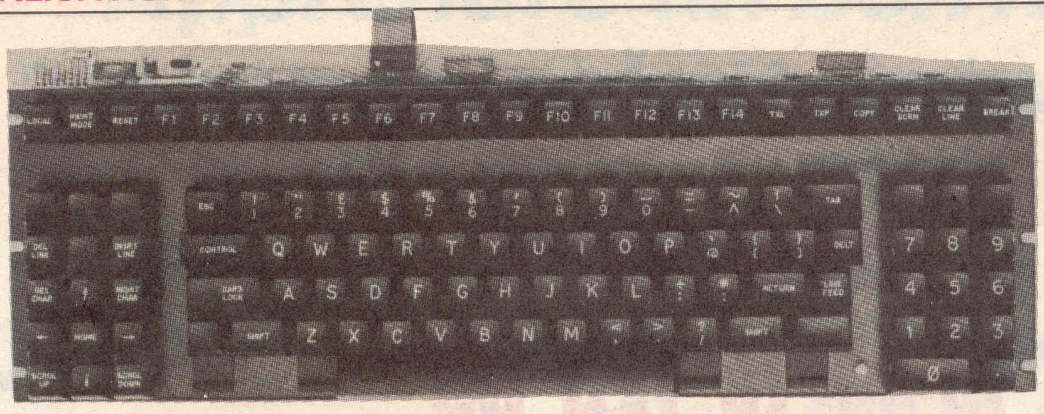


Photo 1. – Le clavier 117 touches proposé pour la version « haut de gamme » de l'ordinateur individuel.

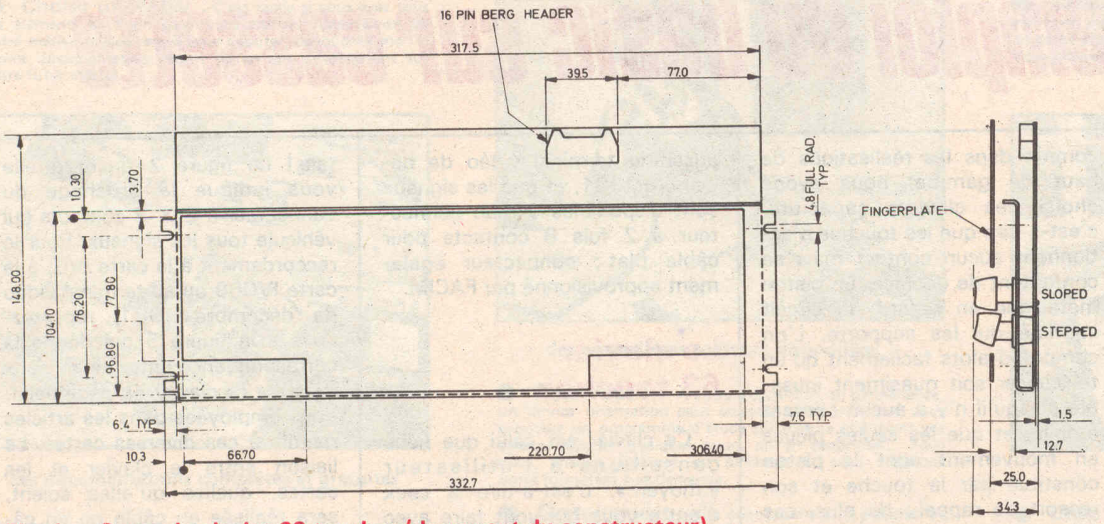


Fig. 1. – Cotes du clavier 63 touches (document du constructeur).

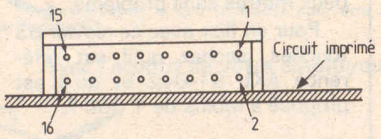


Fig. 2. – Brochage du connecteur du clavier 63 touches.

Patte n°	Signal	Patte n°	Signal
1	BUSY	9	+ 5 V
2	D_0	10	D_4
3	STROBE	11	+ 5 V
4	D_1	12	D_5
5	–	13	Masse
6	D_2	14	D_6
7	Sortie série	15	Masse
8	D_3	16	D_7

NC = non connecté

Fig. 2b. – Brochage du connecteur du clavier 63 touches.

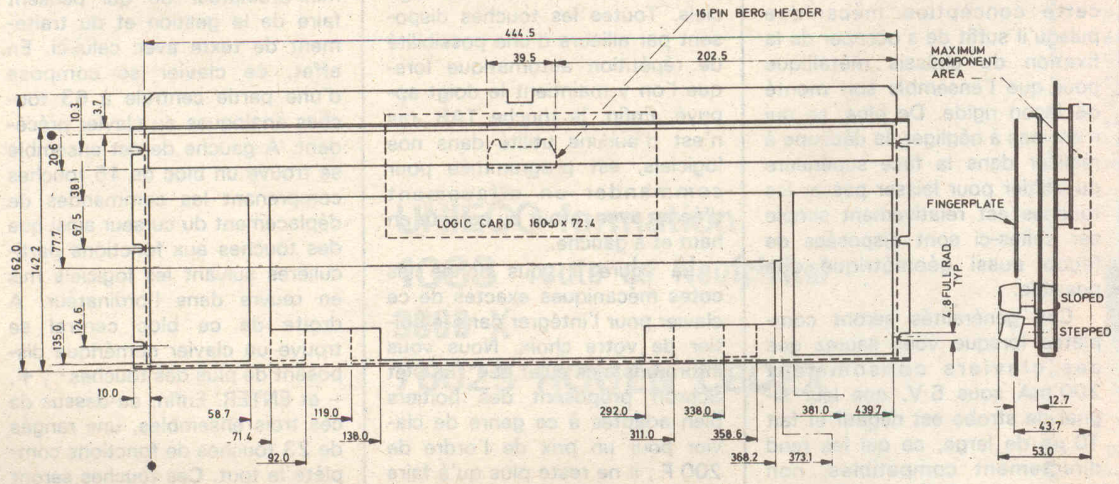


Fig. 3. – Cotes du clavier 117 touches (document du constructeur).

certaines commandes de programmes évolués sans avoir à en frapper le nom comme sur un clavier classique. Une touche ALPHA LOCK est disponible comme sur le clavier précédent et les minuscules sont également disponibles.

La figure 3 vous présente les cotes de ce clavier, si vous souhaitez en réaliser l'intégration vous-même dans un boîtier de votre choix ; nous vous signalons cependant que la société INCODEC propose dès la fin de ce mois un boîtier spécialement conçu pour ce clavier. Nous invitons les personnes intéressées à prendre contact directement avec cette société pour tout renseignement à ce sujet.

Ce clavier (ainsi que le précédent, mais cela présente moins d'intérêt) dispose d'un codage par PROM type 2716 des codes générés par les touches dans les trois modes : SHIFT, NORMAL et CONTROLE. Il va sans dire que cette possibilité est extrêmement intéressante puisqu'elle sera exploitée ultérieurement pour nos futurs logiciels de haut niveau. Comme le 63 touches, la répétition est automatique par maintien d'une touche enfoncée ; de plus, cette possibilité est programmable (au moyen de la 2716) touche par touche, ainsi que la vitesse de répétition qui peut être choisie de 10 ou 20 caractères par seconde.

Pour en finir avec ce clavier, sachez qu'il s'appelle AKL 81-143.

Raccordement des claviers

Celui-ci ne présente aucune difficulté, d'autant que le seul signal qui aurait pu poser des problèmes (le STROBE) n'en pose pas ! La figure 5 vous indique comment relier les signaux en provenance des connecteurs 16 points des claviers, dont les brochages vous ont été indiqués figures 2 et 4, aux connecteurs des cartes IVG, IVG09 (identique à IVG) et terminal vidéo de décembre.

Sauf erreur de câblage, le fonctionnement doit être immédiat dès la liaison établie. Si vous possédez un 117 touches, ne soyez pas surpris du comportement curieux que peuvent engendrer certaines touches non gra-

vées ; celui-ci se normalisera avec les programmes sur disquettes prévus pour exploiter leurs fonctions.

La carte IVG09

Comme annoncé le mois dernier, nous avons étudié une carte de visualisation dérivée de la carte IVG de notre ancien système et parfaitement adaptée à notre ordinateur actuel. Cette carte porte la référence IVG09 et est d'ores et déjà disponible chez FACIM sous forme de circuit imprimé. Son schéma est très proche de celui de notre « ancienne » carte IVG ; en effet celle-ci était très performante et donnait toute satisfaction, hormis sur deux points de détail que nous avons corrigés, à savoir : la vidéo demi-teinte fonctionne maintenant normalement sur IVG09 (le fond reste de la même couleur qu'en vidéo normale et des carrés noirs n'apparaissent plus sur les fins des lignes qui précèdent celles en vidéo demi-teinte), et l'effet de neige observé lors de toutes les sorties de caractères sur l'écran a été supprimé. De plus, il est maintenant possible de monter en générateur de caractères une 2732 (type Intel ou Motorola), ce qui vous permet de disposer de deux jeux de caractères totalement indépendants permettant la réalisation d'applications spéciales (on peut, par exemple, disposer de caractères grecs ou japonais, ou autres...).

Ces améliorations n'ayant pas modifié profondément le schéma de la carte IVG, son circuit imprimé diffère peu de celui de la carte IVG initiale et, si vous possédez la carte IVG initiale et que quelques fils ne vous font pas peur, nous vous indiquerons comment transformer votre IVG en IVG09.

Par ailleurs, les deux cartes IVG et IVG09 étant interchangeables aussi bien pour le nouveau mini-ordinateur que pour l'ancien, FACIM va arrêter progressivement la fabrication des cartes IVG pour ne plus faire que des IVG09.

Compte tenu des délais d'impression du journal d'une part, et des délais de réalisation des films de circuits imprimés d'autre part, nous ne pouvons vous proposer ce mois-ci la publication des films de cette nouvelle carte IVG09 ; nous consacrerons donc notre article complet du mois prochain au schéma et à la réalisation de la carte IVG09. Nous préférons cette solution à celle consistant à vous faire acheter les anciens numéros du Haut Parleur qui ont été consacrés à IVG pour plusieurs raisons :

— Ces numéros ne sont pas en quantité suffisante pour toutes les personnes intéressées par cette réalisation.

— L'étude qui y était faite s'adressait à notre ancien mini-ordinateur ; elle serait donc mal adaptée ici et nous obligerait à faire de nombreuses annotations qui rendraient la lecture de l'ensemble trop confuse à notre goût.

— Le schéma d'IVG avait été publié sur trois numéros différents, ce qui contribue à accroître la confusion évoquée ci-avant.

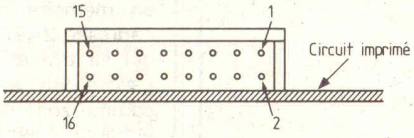
Nous vous demandons donc un mois de patience pour disposer d'une étude d'IVG09 propre et complète. Par contre, pour vous avancer, nous vous indiquons en figure 6 la nomenclature exacte et complète des composants, afin que vous puissiez approvisionner ceux-ci et être prêts à souder dès le 15 octobre ! Attention, la PROM 7611 DECVIS09 est à commander chez FACIM pour une utilisation de IVG09 dans cet ordinateur individuel ; les réalisateurs de l'ancien mini-ordinateur qui souhaitent y mettre IVG09 doivent commander une PROM référencée DECVIS.

Les deux générateurs de caractères doivent vous être programmés par l'auteur. Il faut vous procurer soit deux 2716, soit une 2716 et une 2732 (revoir ce que nous avons déjà écrit au sujet des divers types de 2732 dans nos précédents numéros).

Procédez alors comme pour TAVBUG09 mais en indiquant avec vos mémoires la mention GCGG09. Les frais à prévoir sont les mêmes que pour TAVBUG09. Si vous nous adressez une 2732 pour pouvoir bénéficier du double générateur de caractères, nous laisserons vierge la partie haute de la mémoire afin que vous puissiez vous-même y placer par la suite, au moyen d'un programmeur de PROM (que nous décrirons), les caractères qui vous conviennent.

Enfin, sachez que nous pouvons vous proposer deux types de générateurs de caractères selon ce que vous voulez faire de votre ordinateur. Si vous nous demandez GCGG09, nous vous fournirons un générateur classique correspondant au clavier informatique standard ; tandis que

Photo 2. — Le clavier 63 touches décrit dans ces pages.



Patte n°	Signal	Patte n°	Signal
1	Sortie série	9	D_2
2	8	10	D_3
3	D_7	11	$\overline{\text{STROBE}}$
4	D_6	12	+ 5 V
5	D_5	13	8
6	D_4	14	Masse
7	D_0	15	Masse
8	D_1	16	BUSY

NC = non connecté

Fig. 4. — Brochage du connecteur du clavier 117 touches.

Borne clavier	Borne IVG	Borne Terminal Vidéo
D_0	D_0	DB_0
D_1	D_1	DB_1
D_2	D_2	DB_2
D_3	D_3	DB_3
D_4	D_4	DB_4
D_5	D_5	DB_5
D_6	D_5	DB_6
D_7	D_7	NC
$\overline{\text{STROBE}}$	STROBE*	STROBE
+ 5 V	+ 5 V	+ 5 V
Masse	Masse	Masse
BUSY	NC	NC
Sortie série	NC	NC

* 121 enlevé, pattes 4 et 6 reliées, — S_2 fermé
NC = non connecté

Fig. 5. — Raccordement de ces claviers aux cartes IVG et terminal vidéo.

si vous nous demandez un GCGGA09, nous vous fournirons un générateur de caractères « français » c'est-à-dire disposant du é, du è, du à, du ù, du ç, et des symboles paragraphe (§) et numéro (°), ces caractères étant bien sûr au détriment de caractères rarement utilisés sur un clavier standard ; la table de la figure 7 indique exactement qui remplace qui.

Cette substitution de caractères n'a pas été faite au hasard mais en tenant compte de deux remarques : les caractères supprimés dans la version « française » n'ont pas un rôle « stratégique » en informatique ; la substitution est conforme à celle qu'effectuent les imprimantes EPSON lorsqu'on les programme pour travailler avec le jeu de caractères « français ».

A notre avis, vous avez intérêt à nous demander la version accentuée de ce générateur de caractères, car son seul défaut est minime : lors de l'utilisation du mode d'adressage indirect du 6809, les crochets qui doivent apparaître sur les listings sont remplacés par un « ° » et par un «§ », ce qui ne cause pas une gêne considérable.

Pour ceux d'entre vous qui ne connaissent pas l'ancienne carte IVG, et au risque de nous répéter, il faut bel et bien nous adresser deux 2716 ou une 2716 et une 2732. L'une contient le générateur de caractères dont nous venons de parler ; l'autre, qui est toujours une 2716, contient des constantes utilisées lorsque la carte est en mode graphique.

En résumé donc : pour vous procurer ces mémoires, même procédure que TAVBUG09 en précisant avec vos mémoires GCGG09 pour la version classique ou GCGGA09 pour la version « française ». La participation aux frais demandée étant celle de TAVBUG09 et couvrant juste les frais engagés (emballage, port et temps de programmation),nous n'accepterons pas les mémoires non vierges qui nous ont fait perdre beaucoup trop de temps lors des effacements que nous avons pu réaliser.

Une dernière remarque au sujet des composants d'IVG09 : le quartz à 6 MHz n'est utile que si vous souhaitez utiliser votre carte en mode graphique, et vous pouvez très bien l'employer normalement en alphanumérique sans que ce quartz soit présent.

Nombre	Types et Equivalents	Remarques
1	MC 6845 ou EF 6845	
1	MC 6821 ou EF 6821 ou MC 6820	
12	TMS 4044 ou Intel 2141	FACIM
2	74LS541	
4	74LS242	si 640 sur CPU09
ou 4	74LS243	si 245 sur CPU09
1	8T26	
1	HM7611 pré-programmée	DECVIS09 (FACIM)
2	74LS374	
1	74LS166	
1	74LS163	
3	74LS157	1
1	74121 ou 74LS121	
1	7486 ou 74LS86	
1	74LS10	
1	74LS08	
1	7406 ou 74LS06	
3	74LS00	
2	555 (NE555, LM555, etc.)	
1	2716	Voir texte
1	2716 ou 2732	Voir texte
1	Quartz 16 MHz	
1	Quartz 6 MHz	Facultatif, voir texte
5	1N914 ou 1N4148	
11	Résistances 1/2 ou 1/4 W 5 % carbone 1 × 560 kΩ ; 1 × 270 kΩ ; 1 × 5,6 kΩ ; 1 × 2,7 kΩ ; 1 × 2,2 kΩ ; 2 × 1 kΩ ; 2 × 390 Ω ; 1 × 270 Ω ; 1 × 120 Ω	
2	Condensateurs 10 µF polyester	
1	Condensateur 0,1 µF polyester	
1	Condensateur chimique 0,47 µF 15 V	
2	Condensateurs chimiques 100 µF 10 V	
X	Condensateurs de découplage 22 nF	
1	Pot ajustable 1 kΩ } pas de 2,54 pour CI	
1	Pot ajustable 220 Ω }	
1	Connecteur pour câble plat 34 contacts	FACIM
2	Supports 40 pattes	
2	Supports 24 pattes	
4	Supports 20 pattes	
12	Supports 18 pattes	
7	Supports 16 pattes	
12	Supports 14 pattes	
2	Supports 8 pattes	

Fig. 6. – Nomenclature des composants de la carte IVG09.

Le BASIC sur cassette

Puisque nous consacrerons notre prochain article intégralement à la carte IVG09, nous allons terminer celui-ci par du logiciel en vous présentant le mode d'emploi du BASIC sur cassette que nous vous proposons si vous ne voulez ou ne pouvez pas pousser la réalisation du système jusqu'aux disques souples. Les lignes qui suivent vont vous présenter le mode d'emploi complet du BASIC, c'est-à-dire qu'aucune information ne sera laissée dans l'ombre ; il sera donc inutile de nous écrire pour en savoir plus puisque tout va vous être dit.

Généralités

Le BASIC que nous vous proposons sur cassette ne peut être aussi performant que des BASIC sur disquettes beaucoup plus volumineux ; celui-ci présente cependant quelques particularités intéressantes telles que :

– Très grande vitesse d'exécution.
– Admet le ON ERROR GOTO.
– Dispose du IF THEN ELSE.
– Autorise plusieurs commandes sur la même ligne.
– Dispose d'une fonction USR très puissante.

Ce BASIC sur cassette réside en mémoire de l'adresse 0 à l'adresse 2A85 et utilise la RAM qui se trouve au-dessus de lui, c'est-à-dire à partir de 2A86 comme zone de travail. Tel qu'il est livré, il est initialisé pour utiliser un maximum de 32 K, ce qui est plus que suffisant pour un BASIC sur cassette ; si vous possédez moins de mémoire, ce paramètre peut être modifié. Pour ce faire, chargez la cassette fournie en mémoire puis placez aux adresses 0046 et 0047 (où vous trouvez 7F FF) l'adresse maximum de la RAM contiguë de votre système (3F FF pour 16 K par exemple). Sauvegardez alors sur cassette la mémoire allant de 0 à 2A85 pour disposer d'un

BASIC adapté à votre taille de RAM.

Lorsque votre cassette BASIC est chargée en mémoire, le BASIC peut être lancé à partir de deux points d'entrée. Un lancement à partir de l'adresse 0 est à faire lors de la première utilisation ; il initialise toutes les variables et vide la mémoire de travail ; lorsque l'initialisation est faite, le BASIC affiche PRET sur l'écran du terminal ; il est alors en attente d'une commande. Il est également possible de lancer le BASIC à partir de l'adresse 3 ; cela a pour effet de ne pas initialiser les variables ni la zone de travail, ce qui signifie que si vous étiez en train de travailler et que vous êtes passé sous le contrôle de TAVBUG09 pour une raison ou pour une autre, le fait de relancer le BASIC par un G en 0003 ne détruit pas le programme contenu dans la mémoire du BASIC et vous permet de continuer à l'utiliser comme si de rien n'était.

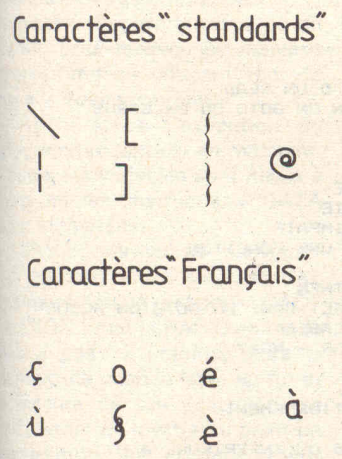


Fig. 7. – Tableau de correspondance entre les caractères « standards » et les caractères « français » dans nos générateurs GCG09 et GCGGA.

Définitions et conventions

Certaines touches ont un rôle particulier, rôle que l'on retrouvera sur tous nos logiciels. Ce sont :

– CNTRL H qui efface le dernier caractère frappé et fait revenir le curseur en arrière d'une position. CNTRL H peut être frappé autant de fois que nécessaire au sein d'une même ligne.

– CNTRL X qui efface la ligne sur laquelle se trouve le curseur ; le « message » X est alors affiché.

– CNTRL C qui permet d'interrompre un programme BASIC lorsque celui-ci attend une entrée de données et fait revenir le BASIC en mode d'attente de commande ; le message BREAK LIGNE XX est alors imprimé, XX étant le numéro de la ligne où est intervenu le CNTRL C.

Pour nos amis lecteurs novices, précisons que CNTRL Z signifie qu'il faut appuyer sur la touche CONTROL et, pendant qu'elle est enfoncée, appuyer sur la touche Z.

Précisons aussi que sur les claviers préconisés ci-avant, CNTRL H peut être remplacé par la touche « flèche vers la gauche » (qui correspond au retour arrière du curseur) tandis que CNTRL X peut être remplacé par DELETE.

Par ailleurs, pour clarifier certaines parties de l'exposé qui va suivre, nous allons adopter deux conventions de notation. Dans les expressions que nous allons présenter, les paramètres indispensables seront représentés entre crochets (< >) tandis que les paramètres facultatifs seront représentés entre parenthèses (( )).

Enfin, et avant de poursuivre, précisons que ce qui va suivre n'est pas un cours de BASIC mais seulement le mode d'emploi de notre BASIC.

Lignes constantes et variables

Le BASIC possède deux modes de fonctionnement, le mode programmé et le mode calculateur. En mode calculateur, chaque ligne frappée est exécutée immédiatement tandis qu'en mode programmé, un ensemble de lignes constituant un programme est exécuté sur commande. La différence entre les deux modes est faite de la façon suivante : le fait de commencer une ligne par un numéro place le BASIC en mode programmé, tandis que le fait de frapper une commande sans numéro de ligne fait exécuter celle-ci aussitôt, le BASIC étant alors en mode calculateur, appelé aussi mode immédiat.

Une ligne est un ensemble de caractères terminé par un retour chariot. Elle peut comporter jusqu'à 127 caractères (et ce même si les lignes de votre terminal ont une capacité inférieure, cela n'a aucune influence).

Il est possible de placer plusieurs instructions BASIC sur la même ligne à condition de séparer celles-ci entre elles par deux points (:). Par ailleurs, les commandes et instructions BASIC peuvent être frappées en majuscules ou minuscules indifféremment. Le BASIC les convertit automatiquement en majuscules. Par contre, dans les chaînes de caractères, le BASIC respecte vos désirs et ne modifie aucunement ce que vous avez frappé, et l'on peut donc travailler en majuscules ou en minuscules sans problème.

Dans un programme, les lignes sont numérotées ; le numéro doit être placé immédiatement en début de ligne sans signe ou espace le précédant. Les numéros doivent être compris entre 1 et 32767 et doivent être uniques. Le fait de frapper deux lignes avec le même numéro ne fait conserver en mémoire que la dernière des deux. Par ailleurs, nous vous rappelons qu'il est d'usage, lors de l'écriture initiale d'un programme, d'écrire les numéros de 10 en 10. Cela permet de rajouter par la suite des lignes intermédiaires que vous auriez pu oublier. Cela est possible car, quel que soit l'ordre de frappe, les lignes sont toujours exécutées dans l'ordre numérique croissant.

Le fait de frapper un numéro de ligne seul suivi d'un retour chariot efface la ligne qui portait ce numéro. Si aucune ligne n'existait sous ce numéro, cette action est sans effet.

Au sein d'une ligne, et après le numéro de début, les espaces sont ignorés et peuvent donc être utilisés comme vous le désirez, ainsi :

– 10 PRINT SIN (X) aura le même effet que

– 10 PRINTSIN(X), mais cette dernière ligne ne rendra pas vos listings particulièrement lisibles !

Le BASIC travaille comme il se doit sur des réels et il peut manipuler tout nombre positif ou négatif compris entre 10 puissance – 38 et 10 puissance 38, ce qui lui confère une précision de six chiffres significatifs (si cela vous semble peu, attendez le BASIC disque qui, lui, travaille sur 16 chiffres significatifs !). Les nombres réels sont frappés de façon classique, il faut seulement prendre soin de remplacer la virgule apprise à l'école par le point. Ainsi 2,5 sera frappé 2.5 pour que le BASIC comprenne.

Il est également possible de fournir au BASIC des nombres en notation scientifique. Le format est normalisé comme pour tous les BASIC. A savoir que, par exemple, 3,456 que multiplie 10 puissance – 5 sera frappé : 3.456E–5. Le 10 puissance est matérialisé par E suivi de la puis-

DIM × (3,2)
Réserve en mémoire le tableau suivant :

× (0,0)	× (0,1)	× (0,2)
× (1,0)	× (1,1)	× (1,2)
× (2,0)	× (2,1)	× (2,2)
× (3,0)	× (3,1)	× (3,2)

Fig. 8. – Définitions d'un tableau à deux dimensions avec un DIM.

1° : () Expression entre parenthèses
2° : ↑ Elévation à une puissance
3° : – Changement de signe
4° : * et / Multiplication et division
5° : + et – Addition et soustraction
6° : = <>, >, <, <=, >= Opérateurs de relation
7° : NOT
8° : AND
9° : OR

Fig. 9. – Priorité relative des opérateurs du BASIC.

sance à laquelle était élevé 10.

Il est aussi possible d'utiliser comme nombre des expressions ; ainsi, si un tiers doit figurer dans un calcul, plutôt que de frapper sa valeur approchée 0.333333, vous pouvez très bien frapper 1/3 et écrire, par exemple : 6 + 1/3 – 2/5. Le BASIC comprendra.

Enfin un autre type de constantes auxquelles vous n'êtes pas, en général, habitués, est constitué par les chaînes de caractères. Ainsi pourrons-nous définir la constante « BONJOUR » ou la constante « HAUT PARLEUR ». Remarquez qu'une chaîne de caractères est constituée par n'importe quel ensemble de caractères compris entre deux guillemets ou entre deux apostrophes.

Nous avons parlé constantes jusqu'à maintenant ; il est évidemment possible de définir des variables. Celles-ci peuvent être du même type que les constantes, auquel cas elles peuvent recevoir un nom constitué par une lettre ou deux lettres, ou une lettre suivie par un chiffre de 0 à 9 ; ainsi : A, XX, A0, BC, D3, E sont des noms de variables corrects ; par contre, 5C ou 8 ne seraient pas admis.

Cette représentation est valable pour les variables numériques. Pour ce qui est des variables chaînes de caractères, les mêmes règles s'appliquent mais leur nom doit être suivi du symbole dollar ($) ; ainsi : A9$, BD$ ou C$ seront des variables chaînes de caractères.

Il existe enfin un dernier type de variables : les variables indicées, qui servent à constituer des tableaux de valeurs. Ces variables répondent aux règles exposées ci-avant mais doivent être dimensionnées avant utilisation et s'utilisent ensuite avec un indice. Ainsi, par exemple, soit l'instruction DIM A (3) ; cela va avoir pour effet de créer 4 variables appelées A(0), A(1), A(2) et A(3).

Le dimensionnement peut être double et l'on peut ainsi créer des tableaux ou des matrices comme montré en exemple figure 8.

Dans un même programme, le même nom peut être employé pour une variable classique et pour une variable indicée ; ainsi A5 sera différent de A5(0) ; par contre, il est interdit de donner le même nom à une variable indicée simple et à une variable indicée double.

Les opérateurs

Il existe en BASIC quatre types d'opérateurs que nous allons étudier successivement. Commençons par les plus classiques qui sont les opérateurs mathématiques.

Ils sont au nombre de 5 : l'addition (+), la soustraction (–), la multiplication (* et non pas ×!), la division (/) et l'élévation à une puissance (^ et non ** comme sur certains BASIC). Quand on les rencontre dans une expression, et sauf s'il y a des parenthèses pour modifier ce qui suit, les opérateurs sont exécutés dans l'ordre de priorité suivant :

– Elévation à une puissance.
– Changement de signe (– devant un nombre).
– Multiplication et division.
– Addition et soustraction.

Viennent ensuite les opérateurs logiques au nombre de trois : NOT, AND et OR.

– NOT réalise le complément bit à bit de la variable spécifiée.
– AND réalise le ET logique entre deux variables.
– OR réalise le OU logique entre deux variables.

Ces opérateurs s'emploient presque exclusivement dans les tests conditionnels ; ainsi pourrons-nous écrire :

– IF A > 0 AND B > 0 THEN GOTO 100 qui signifiera : si A est positif ET si B est positif, aller en 100.

Cet exemple a introduit le troisième type d'opérateurs qui est celui des opérateurs de relation. Ils sont au nombre de six :

– = qui s'utilise sous la forme A = B et qui signifie A égal B.
– < > qui s'écrit A < > B et qui signifie A différent de B.
– < qui s'écrit A < B qui signifie A inférieur à B.
– > qui s'écrit A > B et qui signifie A supérieur à B.
– <= et >= qui s'écrivent A >= B ou A <= B et qui signifient respectivement A supérieur ou égal à B ou A inférieur ou égal à B.

La dernière famille d'opérateurs surprend en général les personnes non habituées à l'informatique car ils permettent de travailler sur les chaînes de caractères. Ce sont la concaténation notée + qui « ajoute » les chaînes de caractères. Ainsi si A$ = « HAUT » et B$ = « PARLEUR », A$ + B$ vaudra « HAUT PARLEUR » ; mais aussi les opérateurs de relation vus ci-avant. Dans ce cas, la comparaison ne peut pas être numérique. Elle est donc alphabétique et permet ainsi de réaliser un classement très simplement. Par exemple, la chaîne « CLAUDE » sera inférieure à la chaîne « MICHEL ».

Tous les opérateurs vus ci-avant peuvent parfois être combinés dans des expressions complexes ; leurs priorités relatives sont alors celles indiquées dans le tableau de la figure 9.

Les commandes

Nous avons vu que le BASIC pouvait fonctionner en mode calculateur ou immédiat, et en mode programmé. Un certain nombre de « fonctions » ne peuvent être exécutées qu'en mode immédiat : ce sont les commandes du BASIC. Ces commandes ne sont pas des instructions mais des ordres relatifs au fonctionnement général du BASIC. Elles sont au nombre de dix, et nous allons les étudier ci-après. Il faut évidemment les frapper sans numéro de ligne et il est donc interdit de les utiliser dans un programme (nous ne voyons pas d'ailleurs ce qu'elles pourraient y faire !).

– CLEAR : met à zéro toutes les variables d'un programme ; cette commande est automatiquement exécutée lors d'un RUN.

– CONT : permet de continuer l'exécution d'un programme qui a été interrompu par une instruction STOP – auquel cas l'on repart sur l'instruction qui suit immédiatement le STOP –, ou suite à l'arrêt d'un programme par un CNTRL C lors d'un INPUT – on repart alors au niveau de cet INPUT. La commande CONT ne

NUMERO	SIGNIFICATION
30	DONNEE INCORRECTE
31	MANQUE DE DONNEES LORS D'UN READ
32	MAUVAISE VALEUR DANS UN ON GOTO OU ON GOSUB
50	COMMANDE INCONNUE
51	CARACTERE ILLEGAL
52	ERREUR DE SYNTAXE
53	FIN DE LIGNE INCORRECTE
54	LIGNE NUMERO 0 INTERDITE
55	NOMBRE DE PARENTHESES IMPAIR
56	REFERENCE INCORRECTE A UNE FONCTION
57	GUILLEMET MANQUANT
58	THEN MANQUANT DANS IF THEN
60	NUMERO DE LIGNE INCORRECT DANS UN GOTO OU GOSUB
61	RETURN SANS GOSUB PREALABLE
62	ERREUR D'IMBRICATION FOR – NEXT
63	ERREUR FATALE
66	RESUME MAL PLACE
72	ERREUR DE MODE DE FONCTIONNEMENT
73	EXPRESSION ILLEGALE
74	VALEUR SUPERIEURE A 255 OU INFERIEURE A 0
75	VALEUR SUPERIEURE A 32767
76	TYPE DE VARIABLE ILLEGAL
77	INDICE SUPERIEUR A LA VALEUR DEFINIE DANS LE DIM
78	ABSENCE DE DIMENSIONNEMENT
80	DEBORDEMENT DE CAPACITE MEMOIRE
81	DEBORDEMENT DANS UN TABLEAU
90	FONCTION USR INDEFINIE
91	APPEL A USR INCORRECT
94	LONGUEUR DE CHAINE DE CARACTERES INCORRECTE
100	EXPRESSION TROP COMPLEXE
101	DEBORDEMENT DE CAPACITE LORS D'UN CALCUL
102	ARGUMENT TROP GRAND
103	DIVISION PAR ZERO
104	NOMBRE TROP GRAND POUR PASSER EN ENTIER
105	ARGUMENT NEGATIF OU NUL DANS UN LOG
106	ERREUR DE CONVERSION LORS D'UN INPUT
107	RACINE D'UN NOMBRE NEGATIF
108	ERREUR DE CONVERSION – NOMBRE TROP GRAND

Fig. 10. – Liste des codes d'erreurs et signification de ceux-ci.

peut faire repartir un programme interrompu par une erreur de même qu'elle ne fonctionnera pas si vous avez modifié le programme entre la cause de l'arrêt et la frappe de CONT.
- EXIT : permet de sortir du BASIC et de passer sous le contrôle de TAVBUG09. Le BASIC n'est pas modifié par cette commande et, si vous le relancez par un G à l'adresse 3, le programme qu'il contenait en mémoire ne sera pas modifié, tandis qu'un G en 0 initialiserait à nouveau la mémoire du BASIC et détruirait son contenu.
- LIST : permet de visualiser les lignes d'un programme. LIST employé seul fait visualiser tout le programme. LIST NN, où NN est un numéro de ligne, fait visualiser la ligne NN et LIST NN-MM fait visualiser depuis la ligne numéro NN jusqu'à la ligne numéro MM.
- LOAD : permet de charger en mémoire un programme BASIC contenu sur cassette ; voyez le paragraphe consacré aux cassettes pour avoir tous les détails relatifs à cette commande.
- NEW : prépare la mémoire du BASIC pour recevoir un nouveau programme en effaçant tout ce qui s'y trouve contenu.
- RUN : lance l'exécution du programme contenu en mémoire. Toutes les variables sont mises à zéro et les instructions DATA sont initialisées.
- SAVE : permet de sauvegarder un programme sur cassette ; même remarque que pour LOAD.
- TRON : met le BASIC en mode pas à pas ; il imprime alors le numéro de chaque ligne au fur et à mesure de son exécution, ce qui permet de mettre au point un programme au comportement imprévu (!).
- TROFF : remet le BASIC en mode normal suite à un TRON.

Les instructions :

Par opposition aux commandes, les instructions peuvent (et doivent) être utilisées dans un programme. Certaines fonctionnent en mode calculateur, d'autres ne fonctionnent qu'au sein d'un programme. Vous comprendrez aisément pourquoi en lisant leur description ci-après :
- GOSUB < numéro de ligne > : le programme continue son exécution au numéro de ligne spécifié, et ce, jusqu'à ce qu'il rencontre une instruction RETURN qui le fait alors revenir à la ligne suivant immédiatement le GOSUB. Ce GOSUB n'est donc rien d'autre qu'un appel à un sous-programme.
- GOTO < numéro de ligne > : le programme continue son exécution à la ligne spécifiée mais c'est, contrairement au GOSUB, définitif, c'est-à-dire qu'il n'y aura pas de retour automatique à la ligne qui suit le GOTO. GOTO est un saut inconditionnel.
- ON < expression > GOSUB < suite de numéros de lignes > : l'expression est calculée et son résultat est tronqué à sa partie entière. Le programme exécute alors un GOSUB à la ligne déterminée comme suit : les numéros de lignes dans la liste ont une position allant de 1 à N s'il y a N numéros. Le numéro sélectionné est celui dont la position dans la liste est égale au résultat de l'expression. Ainsi si A = 4, ON A GOSUB 80, 90, 100, 110, 120 fera exécuter un GOSUB 110 puisque 110 occupe la quatrième position dans la liste.
- ON < expression > GOTO < liste de numéros de ligne > : fonctionne comme le ON GOSUB mais exécute un GOTO à la ligne déterminée au lieu d'un GOSUB.
- ON ERROR GOTO < numéro de ligne > : si, lors de l'exécution du programme, une erreur de numéro de code inférieur à 50 se produit, le programme saute à la ligne spécifiée.
- RESUME < numéro de ligne > : permet de rendre le contrôle au programme principal après l'exécution de la partie de programme déclenchée par un ON ERROR GOTO. Voyez le paragraphe spécialement consacré à ce sujet pour plus de détails.
- RETURN : termine impérativement tout sous-programme appelé par un GOSUB et permet au BASIC de continuer l'exécution par la ligne qui suit le GOSUB ayant appelé le sous-programme.
- IF < expression > GOTO < numéro de ligne > : l'expression est évaluée et, si elle est vraie, le BASIC saute à la ligne spécifiée après le GOTO. Dans le cas contraire le BASIC continue à exécuter normalement le programme à la ligne qui suit le IF GOTO.
- IF < expression > THEN < numéro de ligne > ou < instruction > : fonctionne de la même façon que le IF GOTO si THEN est suivi d'un numéro de ligne ; par contre THEN peut être suivi d'une instruction BASIC qui sera alors exécutée si l'expression est vraie avant que le programme ne continue normalement. Ainsi IF A = 0 THEN PRINT « A est nul » fera imprimer A est nul si A = 0 avant de passer à la ligne suivante, et ne fera rien imprimer du tout si A est différent de 0.
- IF < expression > THEN < numéro de ligne ou instruction >

ELSE < numéro de ligne ou instruction > : fonctionne comme IF THEN mais, dans ce cas, lorsque l'expression est fausse, l'on ne passe pas immédiatement à la ligne suivante, on exécute d'abord ce qui suit le ELSE.

Les instructions d'entrées/sorties

Elles permettent au BASIC de dialoguer avec l'utilisateur du programme, et l'expérience montre qu'elles sont souvent les plus nombreuses dans un programme ; elles sont au nombre de trois.
- INPUT (« chaîne de caractères ») < liste de variables > : cette instruction fait imprimer la chaîne de caractères (si celle-ci existe puisqu'elle est optionnelle) suivie par un point d'interrogation, puis attend autant de variables que spécifié par la liste de celles-ci. Les variables de la liste doivent être séparées entre elles par des virgules et être du type de ce que va répondre l'opérateur ; ainsi, si vous voulez que l'opérateur réponde par une chaîne de caractères, il faudra faire un INPUT A$, par exemple. L'opérateur doit fournir à une commande INPUT autant de variables que ce que vous avez spécifié ; ces variables seront affectées dans l'ordre de votre liste ; elles doivent être frappées séparées par des virgules et terminées par un retour chariot. Le fait de fournir moins de variables que ce qui est demandé fait imprimer par le BASIC un nouveau point d'interrogation en attente des variables manquantes. Le fait de fournir plus de variables que ce qui était demandé fait tout simplement ignorer les variables surnuméraires. Le fait de frapper un CNTRL C en réponse à un INPUT interrompt le programme et rend la main au BASIC.

Voici quelques exemples d'input : INPUT « Quel est votre âge » ; A auquel il faudra répondre par une variable numérique ; INPUT « Une autre partie » B$ auquel il faudra répondre par une chaîne de caractères (généralement OUI ou NON dans un tel exemple), et enfin INPUT A, B, C$, D auquel il faudra fournir dans l'ordre deux variables numériques qui seront affectées respectivement à A et B, une chaîne de caractères qui sera affectée à C$ et enfin encore une variable numérique affectée à D.
- INPUT LINE < nom de variable type chaîne de caractères > : cette commande permet d'entrer une ligne entière comme chaîne de caractères sous le nom spécifié. Une seule variable est admise après INPUT LINE et aucun texte ne peut être imprimé, contrairement à INPUT.
- PRINT (variable, ou ; variable, ou ; chaîne de caractères, ou ; etc.) fait imprimer ce qui suit le PRINT en respectant les règles énoncées ci-après. Si PRINT n'est suivi d'aucune variable, un simple saut ligne sera effectué. La commande PRINT sépare l'écran du terminal (ou le papier de l'imprimante) en 5 zones de 16 caractères. Quand plusieurs variables spécifiées après PRINT sont séparées par des virgules, chaque virgule fait passer à la zone suivante ; ainsi PRINT A, B fera imprimer la valeur de la variable A en position 1 sur l'écran (1re zone) et la valeur de la variable B en position 16 (2e zone), et ainsi de suite. Le fait de frapper plusieurs virgules est autorisé. Ainsi PRINT A,, B fera imprimer B en position 32 (début de troisième zone). Si le nombre de variables spécifié conduit au-delà de la cinquième zone (position supérieure à 64), un retour chariot – saut ligne est automatiquement fait par le BASIC qui continue l'impression sur la première zone de la ligne suivante. Lorsque les variables qui suivent le PRINT sont séparées par des points virgules, elles sont imprimées les unes à la suite des autres sans utilisation des zones définies ci-avant. Ainsi si A = 2, PRINT « La valeur de A est » ; A fera imprimer : la valeur de A est 2 (remarquez que l'espace entre

est et 2 avait été fourni par nos soins dans la définition de la chaîne de caractères). Les variables à imprimer peuvent être placées dans n'importe quel ordre après un PRINT, et des virgules et des points virgules peuvent apparaître sur une même ligne sans que cela cause d'erreur (hormis peut-être dans la présentation de vos résultats si vous n'avez pas fait assez attention !).

Les boucles

Il n'existe en BASIC qu'un moyen de faire des boucles automatiques, c'est en utilisant le classique FOR TO que nous allons étudier.

– FOR < variable > = < expression 1 > TO < expression 2 > (STEP < expression 3 >) : fait exécuter toutes les instructions comprises entre cette ligne et celle contenant un NEXT (voir ci-après) autant de fois qu'il est spécifié par expression 1 et expression 2 selon le principe suivant. La variable spécifiée est la variable de boucle. Elle sert à compter le nombre de tours de boucle réalisés. Sa valeur initiale est fixée par expression 1 et, à chaque tour de boucle, la valeur de la variable est augmentée par expression 3 spécifié après le STEP. Si STEP n'est pas précisé, la valeur prise par défaut pour expression 3 est + 1. La boucle est exécutée tant que la variable de boucle ne devient pas égale ou supérieure à expression 2, dans le cas où expression 3 est positive. Si expression 3 est négative, la boucle continue tant que la variable de boucle ne devient pas inférieure ou égale à expression 2. Quelles que soient les valeurs de expression 1, expression 2 et expression 3, la boucle sera toujours exécutée au moins une fois. Les boucles peuvent être imbriquées les unes dans les autres en nombre illimité (si ce n'est par la taille de la mémoire !), mais il faut alors utiliser des noms de variables de boucles différents. Une boucle peut être quittée prématurément par un GOTO mais il ne faut pas entrer dans une boucle autrement que par le FOR TO initial sinon les résultats sont imprévisibles car la valeur initiale de la variable de boucle n'est alors pas connue.

– NEXT < variable > : est utilisée pour spécifier où se termine une boucle et fait incrémenter la variable de boucle de la valeur spécifiée après le STEP.

Un exemple de boucle très simple :

```
10 FOR I = 1 TO 10
20 PRINT I, I ^ 2
30 NEXT I
```

fera imprimer en position 1 sur l'écran les nombres de 1 à 10 et en position 16 sur l'écran (à cause de la virgule entre I et I 2) leurs carrés (STEP n'ayant pas été précisé, I augmente de 1 à chaque tour de boucle).

Les instructions de fin de programme

Il n'en existe que deux, vu le rôle assez limité de ce genre d'instruction.

– END : termine l'exécution d'un programme lorsque l'on passe sur la ligne qui le contient. Sa présence est optionnelle, le BASIC s'arrêtant alors sur la dernière ligne rencontrée. Un programme terminé par un END ne peut être relancé par une CONT.

– STOP : suspend l'exécution d'un programme et fait imprimer le message : STOP LIGNE XX, où XX est le numéro de la ligne contenant le STOP. Le programme peut être relancé par un CONT ; il part alors de l'instruction qui suit la ligne contenant le STOP.

Les assignations de valeurs

Il existe plusieurs instructions qui permettent de donner à des variables les valeurs de votre choix, et ce, par programme.

– LET < variable > = < expression > : donne à la variable spécifiée la valeur de l'expression. Ce BASIC admet de plus le LET implicite, c'est-à-dire qu'il revient au même d'écrire : LET A = 2 que A = 2.

– DATA < nombre ou chaîne de caractères > (, < nombre ou chaîne de caractères >, etc.) : définit une liste de valeurs qui seront affectées aux variables rencontrées dans les instructions READ décrites ci-après. Le nombre de nombres ou de chaînes de caractères qui suivent DATA n'est pas limité, sinon par la longueur maximum de la ligne autorisée par le BASIC. Un programme peut contenir autant d'instructions DATA que nécessaire ; leurs contenus seront considérés comme un ensemble global dont les éléments seront placés conformément à l'ordre d'apparition des diverses lignes DATA. Il est interdit de placer des DATA dans des lignes comportant plusieurs instructions. Les divers nombres ou chaînes de caractères doivent être séparés par des virgules. Si une chaîne de caractères comporte une virgule, la chaîne complète doit être placée entre guillemets ; dans le cas contraire et si le DATA n'est suivi que par des chaînes de caractères, celles-ci peuvent être écrites sans guillemet ; ainsi : DATA JANVIER, FEVRIER, MARS sera valable.

– READ < variable > (, < variable >, etc.) : est le complément de DATA. Cette instruction affecte à la première variable spécifiée la première donnée rencontrée dans le premier DATA du programme, et ainsi de suite. Le nombre de variables figurant dans un READ peut être inférieur au nombre de données spécifiées dans un DATA, les données surnuméraires seront ignorées. Par contre le nombre de variables spécifiées dans un READ ne doit pas dépasser le nombre de données spécifiées dans l'ensemble des DATA du programme, sinon il y a génération de l'erreur 31. Il faut aussi faire attention à ce que les variables définies après le READ soient du même type que les données qui vont leur correspondre dans les DATA (nombre pour une variable numérique, chaîne de caractères pour une variable chaîne de caractères).

De plus, lors de l'exécution du premier READ d'un programme, le pointeur dont dispose le BASIC pour ces fonctions est remis à zéro et pointe donc sur la première donnée du premier DATA disponible.

– RESTORE : cette instruction remet à zéro le pointeur des données utilisé par les instructions READ ; c'est-à-dire que le premier READ qui va suivre un RESTORE, au lieu de prendre la donnée disponible à la suite dans la liste des DATA, va aller prendre à nouveau la première donnée du premier DATA disponible comme lors de l'exécution du premier READ.

Les instructions diverses

Elles n'entrent dans aucune des catégories précédentes et ce ne sont pas non plus des « fonctions » étudiées ci-après ; nous les avons donc groupées ici.

– DIM < variable 1 > (N ou N,M) (, < variable 2 > (P ou P,Q), etc. : cette instruction a déjà été évoquée en début de ce mode d'emploi, lors de la description des variables indicées. Elle doit être impérativement utilisée pour toutes les variables indicées apparaissant dans un programme et doit être placée avant la première utilisation de la ou des variables concernées.

– POKE < adresses >, < donnée > : place la donnée spécifiée à l'adresse indiquée, ces deux valeurs devant être exprimées en décimal. L'adresse doit être comprise entre 0 et 65535 et la donnée doit être comprise entre 0 et 255. Cette instruction est à employer avec précaution car elle agit directement sur la mémoire, ce qui peut avoir des conséquences fatales si vous touchez par erreur à la zone contenant le BASIC ou ses variables.

– DEF FN < variable > (variable « bidon ») = < expression > : permet de définir autant de fonctions que vous le désirez avec les restrictions suivantes. La « variable » accolée à FN doit être une variable numérique (voir le début de ce mode d'emploi pour les noms autorisés). Ainsi : FNAB ou FNX2 seront des fonctions valides mais pas FNA$ (A$ n'est pas numérique). La variable « bidon » doit être numérique et ne sert qu'à passer un paramètre à la fonction. Un nom de fonction peut être utilisé plusieurs fois dans un programme avec des définitions différentes car seule sa dernière définition est prise en compte. Une fonction définie de cette façon doit l'être en une seule ligne BASIC, doit n'utiliser qu'une seule variable « bidon » de passage de paramètre, et les fonctions utilisant des chaînes de caractères ne sont pas admises. Un exemple :

```
10 DEF FNZZ (X) = X * 2
100 LET X = 10
```

200 Y = 250 + FNZZ (X) donnera à Y la valeur 270 (250 plus 2 fois 10).
– REM (commentaires) : cette instruction n'en est pas une à proprement parler puisqu'elle ne sert qu'à placer du commentaire dans un listing. Le BASIC se limite à reproduire celui-ci intégralement lors d'un LIST mais ne tient jamais compte de REM lors de l'exécution d'un programme. Attention ! REM consomme de la place mémoire, surtout si vous faites comme certains auteurs qui utilisent des REM pour imprimer le mode d'emploi du programme sur le listing !

Les fonctions mathématiques

Elles sont classiques sur tout BASIC digne de ce nom – sauf peut-être l'arc tangente qui n'est pas toujours proposé.
– EXP (X) : fournit l'exponentielle de X, c'est-à-dire « e » (la base des logarithmes népériens, soit 2,718281828) à la puissance X.
– LOG (X) : fournit le logarithme népérien ou naturel de X, c'est-à-dire le logarithme à base « e ». Rappelons que pour passer en logarithme d'une autre base, il suffit de faire LOF (X) en base B = LOG (X)/LOG (B). X doit, bien sûr, être strictement positif.
– SQR (X) : donne la racine carrée de X qui doit être positif ou nul.
– SIN (X) : donne le sinus de X ; X étant exprimé en radians.
– COS (X) : donne le cosinus de X ; X étant exprimé en radians.
– TAN (X) : donne la tangente de X ; X étant exprimé en radians.
– ATN (X) : donne l'arc tangente de X ; la valeur fournie étant toujours comprise entre – PI/2 et + PI/2.
– PI : est la constante PI avec six décimales (3,141692) et peut être utilisée sous ce nom dans les calculs ainsi pour calculer la surface d'un cercle écrirons-nous : LET S = PI * R ∧ 2.
– RND (X) : génère un nombre aléatoire compris entre 0 et 1 qui peut être exploité pour générer un nombre aléatoire sur n'importe quel intervalle en utilisant la formule (M – N) * RND (0) + N ; le nombre ainsi généré sera compris entre N et M. Lorsque X = 0 un nouveau nombre aléatoire est généré à chaque appel de RND (0), c'est l'utilisation normale de cette fonction. Si X est positif, RND (X) fournit le dernier nombre aléatoire qui a été généré. Si X est négatif, un nouveau nombre est généré à chaque appel de RND (X) mais, chaque fois que X prend une valeur déjà utilisée au préalable, le même nombre aléatoire est généré.
– SGN (X) : donne le signe de X sous la forme suivante : SGN (X) est égal à + 1 si X est positif, à – 1 si X est négatif, et à 0 si X est nul.
– ABS (X) : est la valeur absolue de X : ABS (X) = X si X est positif et ABS (X) = – X si X est négatif.
– INT (X) : est le plus grand entier inférieur à X. Pour les nombres positifs, pas de problème, INT (4.3) = 4 ; par contre, attention aux nombres négatifs ; INT (– 5.3) = – 6.

Les fonctions chaînes de caractères

Il est possible d'exécuter certaines fonctions décrites ci-après sur les chaînes de caractères.
– ASC (X$) : donne le code ASCII du premier caractère de la chaîne X$. Un 0 est fourni si la chaîne est la chaîne nulle, c'est-à-dire ne contient aucun caractère.
– CHR$ (I) : est l'inverse de ASC puisqu'elle fournit le caractère dont le code ASCII est égal à la valeur de I. I doit donc être compris entre 0 et 255.
– HEX (X$) : convertit la chaîne de caractères X$, supposée être de l'hexadécimal, en son équivalent décimal ; ainsi : HEX (« 80 ») donnera 128 puisque 80 hexadécimal est égal à 128 en décimal.
– LEFT$ (X$, I) : prélève dans la chaîne X$ les I caractères de gauche ; ainsi si X$ = « HAUT PARLEUR », LEFT$ (X$, 4) = « HAUT ».
– LEN (X$) : donne le nombre total de caractères de la chaîne X$. Tous les caractères sont comptés, y compris les espaces.
– MID$ (X$, I, J) : prélève dans la chaîne X$ les caractères compris entre les positions I et J pour former une nouvelle chaîne de caractères ; ainsi si X$ = « HAUT PARLEUR », MID$ (X$, 5, 8) = « PAR ».
– RIGHT$ (X$, I) : prélève dans la chaîne X$ les I caractères de droite pour former une nouvelle chaîne ; ainsi avec notre X$ précédent, RIGHT$ (X$, 7) = « PARLEUR ».

Pour les trois fonctions LEFT$, RIGHT$ et MID$, I doit être positif ou nul, et inférieur à 32767, sinon il y a génération de l'erreur 74.
– STR$ (X) : fournit une chaîne de caractères qui représente la valeur numérique de X ; ainsi, si X = 123, STR$ (X) = « 123 », ce 123 étant maintenant une chaîne de caractères.
– VAL (X$) : réalise l'inverse de STR$ et convertit la chaîne de caractères X$ en sa valeur numérique. VAL (X$) = 0 si le premier caractère de la chaîne autre qu'un espace est autre chose qu'un signe + ou – ou qu'un nombre.

Les fonctions diverses

Nous n'avons pu les classer ailleurs ; elles sont au nombre de cinq.
– PEEK (I) : donne le contenu décimal de la mémoire d'adresse spécifiée en décimal par I. I doit être compris entre 0 et 32767, sauf si la fonction HEX est utilisée, auquel cas I peut atteindre 65535. La valeur fournie par PEEK est comprise entre 0 et 255.
– POS : indique la position de la tête d'impression de l'imprimante ou du curseur du terminal. Les valeurs commencent à 0. Ainsi si POS = 0, le curseur se trouve sur la première colonne.
– SPC (I) : cette fonction ne doit être utilisée que lors d'un PRINT et fait imprimer I espaces sur le terminal ou l'imprimante.
– TAB (I) : cette fonction ne doit être utilisée que lors d'un PRINT et a pour effet de déplacer le curseur ou la tête de l'imprimante sur la colonne I. Si la colonne est déjà dépassée, cette commande est ignorée. I doit être positif et inférieur à 256.

Utilisation du ON ERROR GOTO

ON ERROR GOTO est en fait une fonction qui peut être validée ou non à tout instant dans un programme. L'activation de cette possibilité a lieu en plaçant dans un programme ON ERROR GOTO < numéro de ligne > ; dans cette condition, toute erreur de code inférieur à 50 (strictement) fera sauter le programme à la ligne spécifiée pour y exécuter l'ensemble d'instructions s'y trouvant. Cet ensemble d'instructions sera impérativement terminé par un RESUME qui fera alors reprendre l'exécution, soit au niveau de ce qui a causé l'erreur, soit à la ligne spécifiée après le RESUME, car l'on peut aussi écrire RESUME (numéro de ligne).

Le programme de traitement de l'erreur peut faire appel à deux variables positionnées par le BASIC pour savoir ce qui s'est passé. Ces deux variables sont ERR et ERL. ERR est égale au numéro de code de l'erreur tandis que ERL est égale au numéro de la ligne de programme ayant causé l'erreur.

La fonction ON ERROR GOTO peut être désactivée à tout instant dans un programme en écrivant simplement ON ERROR GOTO ou ON ERROR GOTO 0. Dans ces conditions, toutes les erreurs qui suivent provoqueront l'arrêt du programme et l'impression d'un message d'erreur.

Utilisation des commandes LOAD et SAVE

Ces deux commandes permettent respectivement de charger et de sauvegarder un programme à partir d'une cassette. L'utilisation en est très simple. Pour charger une cassette en mémoire du BASIC, positionnez celle-ci au début de la partie contenant réellement du signal, frappez LOAD, mettez votre magnétophone en lecture et frappez retour chariot. Lorsque le chargement est terminé, le BASIC affiche à nouveau PRET.

Pour sauvegarder un programme sur cassette, positionnez celle-ci après la bande amorce, frappez SAVE, mettez le magnétophone en enregistrement et frappez alors retour chariot. Le BASIC affiche PRET lorsque l'opération est terminée. Attention ! rien n'est visible sur l'écran du terminal pendant les

LOAD et les SAVE, contrairement à certains de nos anciens BASIC. Si vous éprouvez des difficultés avec ces commandes, revoyez ce que nous avons écrit au sujet de l'azimutage des têtes et de leur nettoyage lors de l'étude de la carte CPU09.

Les erreurs

Le BASIC reconnaît un certain nombre d'erreurs lors de l'exécution d'un programme, et peut vous indiquer celles-ci au moyen d'un code, sous la forme : ERREUR XX LIGNE YY, où XX représente le numéro de code de l'erreur tandis que YY représente le numéro de la ligne où est arrivée l'erreur. La figure 10 vous indique la correspondance entre les codes et leurs significations. Remarquez qu'il n'y a pas de code inférieur à 30, cela par souci de compatibilité avec le DOS qui utilise des numéros d'erreurs compris entre 1 et 30. Nous vous préciserons cela lors de l'étude de celui-ci.

La fonction USR

Cette possibilité du BASIC est à réserver aux programmeurs expérimentés. Elle permet d'appeler un programme en langage machine à partir du BASIC avec passage de paramètre dans les deux sens. Son fonctionnement est strictement conforme à ce qui suit.

La syntaxe est A (ou LET A) = USR (B). Le BASIC évalue alors la variable B et la convertit en un entier sur 16 bits en complément à deux, puis place la valeur obtenue en MEMAX-4. Il va ensuite chercher en MEMAX-2 une adresse et effectue un JSR à cette adresse où il espère trouver un programme en langage machine, sauf si le contenu de MEMAX-2 est nul. Le programme ainsi appelé doit se terminer impérativement par un RTS et doit restituer la pile S dans l'état où elle se trouvait lors de l'appel. De plus, ce programme ne doit pas utiliser plus de 256 octets de pile. Lors du retour au BASIC, celui-ci va lire la valeur contenue en MEMAX-4 (et MEMAX-3 car il prend une valeur sur 16 bits) et donne cette valeur à la variable A de notre exemple précédent.

Dans ces explications, MEMAX n'est autre que la taille maximum de la mémoire, taille qui est contenue en 46 et 47, comme nous l'avons expliqué au début de cette notice.

Vu les possibilités de cette commande USR, il faut l'employer avec circonspection, surtout si vous n'êtes pas sûr de vous, car il est facile d'effacer accidentellement la mémoire contenant le BASIC, ou votre programme, ou les deux si vous avez de la chance !

Une petite astuce

Si vous avez écrit un programme que vous n'avez pas sauvegardé sur cassette (ce qui n'est pas très prudent !) et que, pour une raison quelconque, vous perdiez le contrôle du BASIC, vous pouvez recharger la cassette en mémoire et la lancer à l'adresse 3. Vous récupérerez alors ce qui reste de votre programme.

Dernière minute

Conformément à ce que nous avions annoncé dans notre numéro de juillet, notre système sera présent au SICOB sur le stand du micro club 6809 AFIN—CAU qui a eu l'amabilité de nous inviter. L'auteur sera, bien sûr, présent pour des démonstrations et pour répondre à vos questions. Cependant, au moment où nous écrivons ces lignes (mi-juillet), le programme exact de ces journées n'est pas encore arrêté ; il pourra, par contre, vous être communiqué en téléphonant à la rédaction du Haut-Parleur ou au secrétariat du club AFIN-CAU ((1) 874-38-03) à partir du 13 septembre.

Au sujet des claviers à 63 touches et en raison de l'impatience de certaines personnes, FACIM en a livré avec la 2716 de codage des touches non programmées, comme nous l'avons indiqué ci-avant. Cette modification peut être faite par simple échange de la 2716 qui équipe vos claviers contre une 2716 à la programmation correcte. Toute correspondance à ce sujet doit être adressée à FACIM.

Nous pensons être en mesure de vous annoncer le mois prochain une nouvelle liste de programmes complète à 90 %. Nous vous indiquerons dans ces pages comment vous la procurer, et nous vous demandons de ne pas nous adresser de courrier à ce sujet avant cette date.

Conclusion

Nous en avons terminé avec cet article un peu long, mais qu'il était souhaitable de ne pas fragmenter, et nous vous donnons rendez-vous le mois prochain pour la carte IVG09. ...à suivre...

C. TAVERNIER

Bloc-notes

La table de lecture à bras tangentiel NEC P 560

La table de lecture NEC P 560 est une platine automatique à bras tangentiel, à entraînement direct et stroboscope. Coffret gris clair, tableau de commande gris foncé. Capot « cristal ».

Les commandes : interrupteur de mise sous tension avec témoin LED. Clavier « touches douces » pour les fonctions suivantes : tailles 30 et 17 cm avec 2 témoins LED, vitesses 33 et 45 t avec 2 témoins LED. Avance du bras (2 vitesses), retour du bras (2 vitesses), CUE ing (abaissement et levage du bras). Répétition avec témoin LED. Play/Cut. Potentiomètre de réglage de vitesse avec visualisation stroboscopique.

Caractéristiques techniques : servo-moteur sans friction 4 phases 8 pôles. Plateau aluminium Ø 295 mm. Pleurage et scintillement moins de 0,05 % WRMS. Rapport signal/bruit : + 68 dB (DIN B).

Livrée avec cellule magnétique. Réponse en fréquences : 20 à 20 000 Hz. Bras équilibré statiquement longueur 165 mm. Séparation des canaux : 25 dB à 1 kHz. Force d'appui : 1 à 1,5 gamme pré-réglée.

Dimensions : 350 × 115 × 380 mm.