

Réalisez votre ordinateur individuel

Description et mode d'emploi DU DOS

COMME nous vous l'avons annoncé dans notre précédent numéro, nous allons consacrer les pages qui suivent à la description et au mode d'emploi du DOS. Le sujet étant assez vaste, commençons sans plus tarder.

Mise en service du DOS

Le DOS qui vous est fourni se compose de deux éléments : une PROM 2732 baptisée TAVBUG09 V1.0 et une disquette simple face 40 ou 80 pistes qui supporte le DOS. La PROM TAVBUG09 V1.0 est à mettre en lieu et place de votre TAVBUG09 d'origine qui peut être effacé ou gardé en réserve en cas de problème. Ce changement de PROM ne correspond pas à une fantaisie de l'auteur mais permet d'apporter quelques améliorations au TAVBUG09 initial ; améliorations qui s'avéreront très utiles pour certains logiciels que nous vous présenterons ultérieurement. Vous pouvez mettre en place ce nouveau TAVBUG09 dès sa réception, même si vous ne mettez pas le DOS en service immédiatement. Les seules différences entre TAVBUG09 V1.0 et TAVBUG09 sont les suivantes :

- La commande E (qui permettait de coder un post byte en adressage indexé a été supprimé puisque, si vous possédez le DOS, vous disposez du macro assembleur 6809 et que vous n'en avez donc plus besoin.
- La commande Z a été ajoutée ; elle permet de changer le format d'écran de la carte

IVG09 et fait passer de 20 lignes de 80 caractères à 24 lignes de 80 caractères ; chaque frappe de Z faisant passer alternativement d'un mode à l'autre et vice-versa. Cette commande était indispensable pour certains logiciels futurs. Elle est, bien sûr, sans action au niveau du terminal vidéo qui, lui, est toujours en 16 lignes de 64 caractères. Ce deuxième format d'écran est obtenu au détriment des caractères minuscules descendants (le p, le y, le j, etc.) dont la partie extrême basse est un peu raccourcie.

- La commande de chargement du DOS a été améliorée et elle est maintenant capable de détecter une absence de disquette DOS ou une erreur sur celle-ci.

- Une possibilité supplémentaire d'adressage direct du curseur a également été ajoutée et nous la décrirons en détail en temps utile.

Pour les spécialistes, précisons que cette commande fonctionne comme celle des terminaux TVI 912.

Tous les sous-programmes décrits dans la notice de TAVBUG09 restent identiques à eux-mêmes dans ce « nouveau » TAVBUG09 et les logiciels que vous possédez ou que vous avez pu écrire avec TAVBUG09 sont utilisables sans modification avec TAVBUG09 V1.0. Nous attirons

l'attention des gens qui ont « bricolé » TAVBUG09 sur le fait que certaines modifications structurelles ont été apportées dans TAVBUG09 V1.0 et il nous est impossible de garantir la compatibilité des « bricolages ». Cela est normal, compte tenu du fait que nous avons bien précisé dans la notice de TAVBUG09 que les seuls sous-programmes utilisables étaient ceux que nous décrivions.

Lorsque votre TAVBUG09 V1.0 est mis en place, vérifiez le fonctionnement de quelques commandes et, si tout est bon, vous pouvez charger le DOS. Un défaut à ce niveau est pratiquement impossible car les PROM TAVBUG09 V1.0 sont vérifiées une par une avant expédition.

Le DOS vous est fourni sur disquette certifiée et vérifiée avant expédition. Cette disquette est simple face, quel que soit votre équipement en lecteurs, puisque cela n'a aucune importance comme expliqué ci-après. Par contre, ainsi que nous l'avons expliqué dans les informations 6809, vous avez dû nous préciser si vous vouliez une disquette 40 pistes ou 80 pistes. En l'absence d'indication de votre part, la disquette qui vous est envoyée est une 40 pistes, puisque cela correspond à l'équipement le plus répandu.

Placez cette disquette, munie de son adhésif de protection en écriture, dans votre lecteur numéro 0 ; vérifiez que le commutateur de suppression des protections est dans la position inactive, fermez la porte et frappez X ; le lecteur

doit se mettre en marche et un bruit de tête doit se faire entendre. Au bout de quelques secondes, le message TAVDOS 6809 Version 1.0 apparaît, suivi par un message de la forme DATE (JJ,MM,AA) ? Frappez alors la date sous la forme demandée (04,02,83 pour 4 février 1983 par exemple). Une faute de frappe peut être corrigée par un CNTRL H ou par la frappe de la flèche vers la gauche pour les claviers qui en sont équipés et dont la PROM de codage est correctement programmée. Un retour chariot termine cette frappe et fait apparaître les trois signes « plus » caractéristiques du DOS en attente de commande.

Il peut arriver, de manière tout à fait exceptionnelle, que le DOS ne se charge pas du premier coup ; le message « ERREUR DE CHARGEMENT DOS » est alors affiché, et le contrôle est rendu à TAVBUG09. Si cela se produit, il suffit de frapper X à nouveau pour que tout rentre dans l'ordre. Ce message est, par contre, affiché si vous commettez une erreur réelle telle que : disquette DOS non mise en place correctement, porte du lecteur numéro 0 non fermée, fonctionnement incorrect de votre carte IFD09.

Les généralités du DOS

Nous allons décrire en détail les commandes dont vous disposez mais, au préalable, il nous faut vous présenter quelques généralités valables dans

tous les cas. Tout d'abord, ce DOS est compatible FLEX (marque déposée de Technical Systems Consultants) bien qu'il soit parfaitement adapté à notre système et que tous les dialogues aient lieu en français (correct qui plus est !).

Pour simplifier votre travail, nous avons fait en sorte que les messages d'erreur qui peuvent apparaître lors du fonctionnement du système soient présentés en minuscules lorsque c'est le DOS lui-même qui les a générés, et en majuscules lorsque c'est une commande du DOS qui les a générés. Cela vous semble peut-être un peu nébuleux pour l'instant, mais vous comprendrez mieux de quoi il s'agit après quelques manipulations et vous serez alors à même d'apprécier.

Les commandes données au DOS peuvent être frappées en majuscules ou en minuscules indifféremment ainsi que les noms de fichiers que le DOS met automatiquement en majuscules. Toute faute de frappe peut être corrigée par la frappe d'autant de CNTRL H ou de flèches vers la gauche que nécessaire, chaque frappe effaçant le caractère précédant le curseur et faisant ensuite reculer celui-ci d'une position (sauf modification de cette fonction par la commande TTYSET décrite ci-après). Une ligne entière peut être effacée par la frappe d'un CNTRL X ; le DOS annule alors son contenu et remplace les trois signes plus par trois points d'interrogation.

Il faut ensuite savoir que toute information contenue sur la disquette est placée dans un fichier. Un fichier peut donc contenir du texte tel que programme en n'importe quel langage, texte quelconque ou autre, ou des informations considérées comme étant du binaire telles que programme à charger en mémoire, contenu de PROM à programmer ou toute autre chose. Les fichiers vous sont connus et accessibles uniquement par leur nom, et vous n'avez pas à savoir où ils sont placés sur la disquette ; c'est le DOS qui s'en occupe, et il fait cela au mieux des possibilités offertes. Pour vous y retrouver, les fichiers portent un nom suivi par un suffixe ; ce nom comporte de

un à huit caractères à choisir parmi les lettres de A à Z et les chiffres de 0 à 9 ainsi que le tiret. Seule restriction : un nom de fichier ne doit pas commencer par un chiffre. Ce nom est séparé de son suffixe, aussi appelé extension, par un point. Le suffixe se compose impérativement de trois lettres (le nombre est obligatoire contrairement au nom où il pouvait aller de un à huit). Ces trois lettres peuvent être quelconques mais, vu la quantité d'information qu'il est possible de ranger sur une disquette, certains suffixes ont été normalisés, et il est conseillé de s'y tenir, au moins pour deux raisons : certaines commandes du DOS admettent que l'on ne précise pas de suffixe (ce qui allège la frappe) mais elles choisissent alors le suffixe normalisé logique, compte tenu de la commande exécutée. Le fait d'utiliser les suffixes normalisés permet de savoir instantanément à quel type de fichier l'on a affaire, et c'est très agréable. Ces suffixes sont les suivants :

- TXT pour tous les fichiers contenant du texte (source d'un programme en assembleur, courrier divers, etc.).
- BIN pour tout fichier contenant du binaire chargeable en mémoire.
- BAS pour tout fichier contenant un programme BASIC destiné à l'interpréteur BASIC sur disquette.
- CMD pour tout fichier considéré comme une commande du DOS.
- SYS pour tout fichier « système » ; le DOS lui-même est ainsi contenu dans un fichier muni de cette extension (TAVDOS09.SYS).
- BAK pour tous les fichiers de sauvegarde créés par l'éditeur lors de l'édition d'un fichier déjà existant (voir mode d'emploi de l'éditeur).
- OUT pour tous les fichiers créés par la commande O (lettre O) décrite ci-après.

Bien que cette notion de nom de fichier flanqué d'une extension soit déjà très souple et précise, elle ne suffit pas dans tous les cas. En effet, lorsque vous parlez d'un fichier, il faut encore préciser le lecteur sur lequel il se trouve. Cette précision est donnée par

le numéro du lecteur (0 à 1, voire 2 si vous avez trois lecteurs). Ce numéro est placé avant ou après le nom de fichier mais en est séparé par un point comme l'extension présentée ci-avant. La présence de ces trois informations : NOM.EXTENSION.NUMERO définit exactement le fichier. Il n'est cependant pas toujours utile de donner les trois, vu la commande demandée au DOS, et il est alors possible de ne fournir que la partie nécessaire, les seules contraintes étant : que le séparateur entre deux éléments du nom de fichier soit un point, et que le numéro de lecteur ne soit pas intercalé entre le nom et l'extension. La figure 1 donne quelques exemples de noms de fichiers qui sont tous corrects.

Une autre notion fondamentale à connaître est le comportement du DOS vis-à-vis du format des disquettes. Lors de la première utilisation d'une disquette, il faut lui faire subir une opération appelée formatage qui consiste à définir les pistes et les secteurs dont nous avons parlé lors de l'étude théorique de celles-ci. Cette opération est faite au moyen de la commande FORMAT et, sous réserve que vous ayez le lecteur adéquat, cette commande peut faire des disquettes 35, 40 ou 80 pistes, simple ou double face. Ces informations de taille et de face sont codées en un point particulier de la disquette, ce qui fait que, à tout instant, le DOS sait sur quel type de disquette il a à travailler. Cela vous décharge de tous soucis puisque, si vous mettez une disquette double face dans un lecteur, le DOS le saura et utilisera bien les deux faces, alors que si vous mettez une disquette formatée en simple face dans un lecteur, le DOS le saura aussi et n'utilisera qu'une face. Cela vous permet de comprendre

pourquoi le DOS est fourni sur une disquette simple face, quel que soit votre équipement. En effet, il fonctionnera ainsi dans tous les cas (que vous ayez des lecteurs simple ou double face). Par contre, lorsque vous allez faire votre disquette système propre (celle fournie devant être conservée à l'abri en cas de « coup dur »), il vous suffira de formater une disquette en double face si tel est votre équipement et de copier la disquette fournie dessus pour avoir un DOS double face. On peut difficilement faire plus simple puisque, une fois que vous aurez compris ces explications, vous aurez aussi compris que vous n'avez plus à vous soucier d'aucune notion de format de disquette ; c'est le DOS qui s'en occupe. Attention, cette fonction intéressante est cependant limitée par deux facteurs : une disquette double face ne pourra pas être lue dans un lecteur simple face, et il y aura génération d'un message d'erreur lorsque le DOS essaiera d'accéder à la seconde face ; et une disquette 40 pistes ne peut être lue dans un lecteur 80 pistes et réciproquement. Ces limitations ne sont pas dues au logiciel DOS mais aux lecteurs de disquettes eux-mêmes. Elles sont d'ailleurs logiques et évidentes, mais l'on s'habitue tellement à être pris en charge par le DOS que l'on arrive parfois à les oublier.

La frappe des commandes

Une commande peut être frappée dès lors que les trois plus (ou trois points d'interrogation si vous avez fait au préalable un CNTRL X) sont visibles. La frappe peut avoir lieu en majuscules ou minuscules et CNTRL H et (ou) CNTRL X peuvent être utilisés à chaque instant comme vu ci-avant. Le DOS ne commence à interpréter la commande qu'après la frappe d'un retour chariot qu'il considère comme étant l'ordre d'exécution de la ligne de commande. Plusieurs commandes peuvent être placées sur une même ligne à condition de les séparer les unes des autres par

```
O.PROG.BIN
PROG.BIN.O
PROG.O
O.PROG
1.PRO-12.TXT
PRO-12
O.PROGRAMM.XYZ.
```

Fig. 1. - Quelques noms de fichiers corrects.

deux points (:). Une ligne de commande ne doit pas comporter plus de 128 caractères ; tous les caractères surnuméraires sont tout simplement ignorés par le DOS.

Lorsqu'une commande nécessite plusieurs éléments, par exemple le nom de la commande suivi par un ou plusieurs noms de fichiers sur lesquels elle doit agir, il faut séparer ces divers éléments par un espace ou une virgule au choix (ces deux séparateurs peuvent coexister pour la même commande sans problème). Dans les exemples que nous donnerons, nous ferons un large usage de la virgule car, lors du passage de notre « manuscrit » à l'imprimerie, l'espace devient sujet à caution !

Une commande est lancée en frappant son nom (noms que nous allons étudier ci-après), ce qui a pour effet de charger le fichier portant ce nom suivi par l'extension CMD en mémoire et de lancer son exécution. Ainsi, lorsque vous frappez la commande DATE pour avoir la date courante, le fichier DATE.CMD, qui n'est autre qu'un programme, se charge en mémoire et s'exécute automatiquement. Cette notion nécessite une précision supplémentaire. En effet, sur quel lecteur le DOS va-t-il aller chercher le fichier de commande ainsi spécifié ? Pour répondre, il nous faut introduire ici la notion de disque « système » et de disque de « travail ». Un lecteur est considéré comme étant le lecteur « système » lorsqu'il contient le disque sur lequel se trouvent toutes les commandes du DOS. A la mise sous tension, c'est au lecteur numéro 0 qu'est alloué ce rôle, et c'est pourquoi nous vous avons fait placer la disquette DOS dans ce lecteur en début d'article. Une commande permet ensuite de changer cela, comme nous allons le voir. Donc, et sous réserve de ne pas avoir exécuté la commande évoquée ci-avant, chaque fois que vous allez frapper un nom de commande, le DOS ira chercher le fichier correspondant sur le disque 0. Si vous voulez faire chercher la commande sur un autre disque, soit vous redéfi-

nirez le disque système grâce à la commande adéquate, soit vous frapperez N.COMMANDE où N est le numéro du lecteur où vous voulez aller chercher la commande ; on peut difficilement faire plus souple d'emploi !

Cela étant vu, et avant de vous présenter les commandes disponibles, nous allons vous préciser les conventions adoptées ci-après pour leur description, conventions analogues à celles employées pour le BASIC et l'éditeur déjà étudiés. Un paramètre indispensable à la commande sera représenté entre deux signes de comparaison (< et >) tandis qu'un paramètre optionnel sera compris entre parenthèses ().

Enfin, et avant de passer aux actes, nous vous rappelons que le meilleur mode d'emploi ne saurait en aucun cas remplacer des manipulations faites au fur et à mesure de sa lecture. Alors, ne vous gênez pas, mais lisez au préalable le paragraphe suivant où il est dit comment faire une copie de sauvegarde de votre disquette DOS ; vous serez après cela tout à fait tranquille, même en cas de « grosse bêtise ».

Sauvegarde de votre disquette DOS

Comme pour tout logiciel dont vous pouvez être amené à faire l'acquisition, il est indispensable de faire immédiatement une copie de l'original qui vous est fourni, afin que vous puissiez en disposer en cas de problème ultérieur. Cela est particulièrement justifié pour le DOS, avec lequel il vous est possible en quelques secondes d'effacer toute la disquette sans aucune difficulté ! Nous vous recommandons donc de suivre, même sans chercher à tout comprendre, les indications ci-après, et, lorsque vous serez en possession de votre disquette copie, vous pourrez alors commencer les manipulations sur le DOS.

Si vous avez deux lecteurs :
 – Placez la disquette DOS origine en 0 et une disquette vierge en 1.

– Formatez la disquette vierge en simple ou double face et en 40 ou 80 pistes, selon votre équipement. Donnez-lui le nom et le numéro de volume que vous désirez (voir ci-après si nécessaire la notice de la commande FORMAT).

– Faites ensuite un COPY 0,1,.SYS,.CMD,.LOW qui va avoir pour effet de recopier toute la disquette DOS sur la disquette que vous avez formatée.

– Faites, lorsque COPY a terminé, un LINK 1.TAVDOS09.SYS.

– Vous pouvez alors sortir la disquette du lecteur numéro 1 et placer celle qui était contenue dans le lecteur 0 à l'abri. La disquette que vous venez ainsi de réaliser est une copie conforme de la disquette que vous avez reçue et vous pouvez l'utiliser comme disquette DOS sans aucune restriction.

Si vous n'avez qu'un lecteur :

– Formatez une disquette vierge comme indiqué ci-après dans la partie de la notice de la commande FORMAT relative aux possesseurs d'un seul lecteur.

– Enlevez la disquette formatée du lecteur, mettez-y à nouveau la disquette DOS d'origine et faites un COPYSD *.

– Suivez alors les indications données, en français, par la commande COPYSD (vous pouvez aussi lire la notice de cette commande ci-après si vous avez un doute sur le comportement à avoir).

– Lorsque tous les fichiers ont été copiés sur la disquette de sauvegarde, laissez-la en place dans le lecteur et faites un LINK 0.TAVDOS09.SYS.

– Vous disposez alors d'une copie conforme de votre disquette DOS d'origine que vous pouvez mettre à l'abri.

Remarque

Pour les manipulations qui précèdent et pour celles que vous pouvez avoir à faire par la suite, nous vous rappelons que les disquettes peuvent être protégées en écriture au moyen d'un adhésif obturant la découpe faite sur un flanc de la disquette. Il est impossible au DOS d'écrire sur une disquette

ainsi protégée, quoi que vous fassiez. N'oubliez donc pas d'enlever ou de ne pas mettre cette protection lorsque vous avez à écrire sur une disquette. Nous vous rappelons aussi que, si vous ne voulez pas passer votre temps à coller et décoller cet adhésif, le commutateur rotatif situé en face avant permet de supprimer cette protection sur le lecteur de votre choix. De toute façon, un message d'erreur vous indique toute tentative d'écriture sur une disquette protégée, sauf dans le cas de la commande FORMAT qui se contente de vous signaler que le formatage a été interrompu (ce qui est normal puisqu'elle n'a pas pu écrire sur la disquette).

Si votre terminal va trop vite

Nous vous rappelons que TAVBUG09 dispose de la possibilité d'arrêter immédiatement toute sortie de caractère lors de la frappe d'une touche quelconque du clavier. Le DOS utilisant TAVBUG09, cette possibilité est conservée (elle peut même vous faire croire parfois à un blocage si vous frappez accidentellement une touche au mauvais moment !); mais, de plus, le DOS dispose d'une possibilité analogue et plus puissante. Lorsque vous avez arrêté une sortie de texte sur le terminal en frappant une touche quelconque, vous avez trois possibilités :

– Vous frappez une touche autre que ESCAPE (ESC sur certains claviers) ; dans ce cas, votre sortie reprend où elle était interrompue.

– Vous frappez ESCAPE ; dans ce cas, rien ne se passe mais c'est maintenant le DOS et non plus TAVBUG09 qui est informé de l'arrêt du listing ; dans ce cas, vous pouvez alors frapper à nouveau ESCAPE et la sortie reprendra comme si de rien n'était, ou vous pouvez frapper un retour chariot, auquel cas la commande qui produisait cette sortie sera prématurément terminée et le contrôle sera rendu au DOS.

Attention, pour un fonctionnement correct de ces possibilités, il ne faut pas être plus

rapide que le terminal ; ainsi, si vous frappez une touche pour arrêter le listing et que vous frappez ESCAPE juste après sans avoir attendu cet arrêt, il se peut fort qu'une de vos deux touches soit ignorée, voire que les deux commandes données trop vite annulent leur effet ; essayez et vous comprendrez ce que nous voulons dire.

Classification des commandes

Dans notre système muni du DOS, trois types de commandes vous sont offertes. Les commandes de TAVBUG09 relatives au DOS :

– La commande X qui permet de charger le DOS à la mise en marche du système ou lorsque vous souhaitez commencer à travailler avec le DOS même si le système est déjà en marche depuis un certain temps. Cette commande est aussi à utiliser lorsque, à la suite d'une erreur (essai d'un programme qui n'est pas au point, par exemple, vous détruisez le DOS qui avait été chargé en mémoire. Il est alors prudent de faire un RESET avant de frapper X car, selon ce que votre programme avait fait du pointeur de pile du 6809, ce chargement peut alors s'avérer impossible.

– La commande Y permet de passer sous le contrôle du DOS (qui est supposé être déjà chargé en mémoire) alors que vous êtes sous le contrôle de TAVBUG09. Elle s'utilise, par exemple, lorsque, pour une raison ou pour une autre, vous avez quitté le DOS pour passer sous TAVBUG09 (pour essayer un programme en pas à pas,

par exemple) et que vous souhaitez ensuite revenir sous le contrôle de celui-ci. Attention, cette commande ne fonctionne correctement que sous réserve que, lors de vos travaux sous TAVBUG09, vous n'ayez pas été touché à la zone mémoire contenant le DOS.

Vous disposez ensuite, au niveau du DOS, de deux familles de commandes : les commandes résidentes, c'est-à-dire celles qui sont chargées en mémoire en permanence en même temps que le DOS, et les commandes transitoires qui sont lues sur le disque système et chargées en mémoire uniquement au moment de leur exécution. La majorité des commandes du DOS est de ce dernier type car cela conduit à une économie considérable au niveau de l'encombrement mémoire. Les deux seules commandes résidentes sont MON et GET (voir ci-après).

Problèmes de compatibilité

Ainsi que nous l'avons déjà maintes fois écrit, les messages du DOS sont intégralement en français. Pour ce qui est des commandes, par contre, nous leur avons laissé leur appellation d'origine sur la disquette DOS que vous avez reçue, pour plusieurs raisons :

- Ce nom est relativement standardisé au niveau des divers DOS du marché actuel.
- Ces appellations sont familières aux réalisateurs de notre « ancien » système.
- Enfin, il vous est facile, grâce à une des commandes décrite ci-après, de changer l'appellation d'origine de n'im-

porte quelle commande par n'importe quel nom à votre convenance.

Pour les mêmes raisons, et sauf très rares exceptions qui seront précisées lorsque ce sera nécessaire, les questions posées par les commandes et auxquelles il faut répondre par oui ou par non comprennent aussi bien l'anglais que le français et vous pouvez répondre O (pour oui) ou Y (pour yes) ou N (pour non ou no). Ces précisions étant faites, abordons l'étude des commandes par ordre alphabétique.

Commande APPEND

Cette commande est utilisée pour mettre bout à bout plusieurs fichiers. Elle peut agir sur des fichiers de n'importe quel type, mais tous les fichiers mis bout à bout doivent, par contre, être d'un seul et même type. Tous les fichiers utilisés par cette commande sont laissés intacts sur le disque. Si des fichiers binaires comportaient une adresse de transfert, c'est celle du dernier fichier de la liste qui deviendra l'adresse de transfert du fichier global. La syntaxe est la suivante :

– APPEND <LISTE DE FICHIERS>, <FICHER GLOBAL>. La liste de fichier pouvant être réduite à deux fichiers comme pouvant comporter N fichiers (sans toutefois que leur énumération dépasse les 128 caractères par ligne de commande autorisés par le DOS). Le nom de fichier qui sera donné à la mise bout à bout des autres est baptisé ci-avant FICHER GLOBAL ; ce

nom de fichier ne doit pas exister déjà sur le disque utilisé ; sinon une question demandant l'autorisation de l'effacer sera posée. Le fait d'interdire cet effacement terminera prématurément la commande. Les noms de fichiers n'ont pas à être précisés en entier ; l'extension prise par défaut est TXT et le lecteur par défaut est celui de travail (voir ci-après la commande ASN). Si une extension est spécifiée pour le premier nom de fichier de la liste, elle devient l'extension par défaut de toute la ligne. Exemple d'utilisation : APPEND CHAPIT1, CHAPIT2, CHAPIT3, 0.LIVRE. Cette ligne aura pour effet de mettre bout à bout les fichiers baptisés CHAPIT1.TXT, CHAPIT2.TXT et CHAPIT3.TXT pris sur le disque de travail et d'appeler l'ensemble ainsi constitué LIVRE.TXT ; ce dernier fichier étant alors placé sur le lecteur numéro 0.

Commande ASN

Cette commande est utilisée pour définir quel va être le disque système et quel va être le disque de travail. Elle permet aussi de savoir qui est présentement le disque système et le disque de travail. Lors du chargement du DOS, les disques système et travail sont tous deux le disque numéro 0. Si vous n'avez qu'un lecteur, cette commande ne vous est donc d'aucune utilité. La syntaxe en est la suivante :

– ASN (W=<NUMERO>) (,S=<NUMERO>) ou ASN (T=<NUMERO>), (,S=<NUMERO>) où NUMERO est le numéro du lecteur que

CATALOGUE DU DISQUE NUMERO 1
DISQUE : PROBAS #0

NOM	TYPE	TAILLE	PROTECTION
GROSC	.BAS	5	WD
GROS-IMP	.BAS	5	
GROSC	.DAT	73	
ETIQDISK	.BAS	10	
ETICAS	.BAS	8	
SEQRAN	.BAS	3	W
LISTZX00	.TXT	3	

SECTEURS DISPONIBLES = 673

Fig. 2. – Exemple d'exécution d'une commande CAT.

```
+++ASN
LE DISQUE SYSTEME EST LE #0
LE DISQUE DE TRAVAIL EST LE #1
+++ASN W=0,S=1
+++ASN
LE DISQUE SYSTEME EST LE #1
LE DISQUE DE TRAVAIL EST LE #0
+++0.ASN S=0,W=1
+++ASN
LE DISQUE SYSTEME EST LE #0
LE DISQUE DE TRAVAIL EST LE #1
```

Fig. 3. – Exemple d'exécution de la commande ASN.

vous voulez affecter au système (S) ou au travail (T pour le français et W pour l'anglais). Si rien n'est spécifié derrière ASN (tout est en effet facultatif puisque nous avons mis des parenthèses), la commande indique alors l'état dans lequel est le DOS, comme montré figure 3.

Commande BUILD

Cette commande permet de fabriquer (la traduction exacte de BUILD étant construire) très rapidement des fichiers texte. C'est un mini-éditeur très peu puissant mais bien utile lorsque l'on veut faire un fichier texte qui ne comporte que quelques mots. La syntaxe en est :

– BUILD <NOM DE FICHIER> où NOM de FICHIER représente le nom qui sera donné au fichier ainsi réalisé. Ce nom n'a pas besoin d'être donné en entier ; l'extension prise par défaut sera TXT et le numéro de lecteur pris par défaut sera celui de travail. L'on peut ainsi frapper BUILD TOTO sur le lecteur de travail, mais rien n'interdit de frapper BUILD 0.PROG.BAS qui créera ainsi le fichier PROG.BAS sur le lecteur numéro 0.

La frappe de cette commande fait imprimer un signe égal en début de ligne, indiquant que BUILD est prêt à accepter votre texte. Celui-ci peut alors être frappé et corrigé au moyen de CNTRL H et CNTRL X (voir ci-avant le rôle de ces touches) étant entendu qu'une fois que le retour chariot de fin de ligne est frappé, la ligne est stockée sur disque et ne peut plus être modifiée (BUILD n'est pas un éditeur !). Pour terminer la commande, il suffit de frapper un dièse (#) immédiatement après le signe égal suivi par un retour chariot. Si, lors de la frappe du texte, vous effacez une ligne par un CNTRL X, ce n'est pas un signe égal qui apparaît sur la ligne suivante mais les trois points d'interrogation du DOS ; ce phénomène est normal et vous pouvez continuer à frapper ce que vous désirez, vous n'êtes pas sorti de la commande BUILD.

Commande CAT

Cette commande permet de visualiser le catalogue d'un disque, c'est-à-dire la liste des fichiers qu'il contient ainsi que leur taille en nombre de secteurs et la protection dont ils bénéficient. Les informations

vous sont données selon la présentation visible figure 2. La syntaxe est la suivante : CAT (NUMERO), (LISTE DE FICHIERS) où NUMERO est le numéro du lecteur dont vous voulez avoir le catalogue ; s'il n'est pas précisé, le lecteur de travail est pris par défaut. LISTE DE FICHIER peut être une liste de noms de fichiers ou d'extensions ou de morceaux de noms de fichiers. Si cette liste existe, le catalogue ainsi fourni sera limité aux seuls fichiers répondant aux contraintes de la liste, ainsi : CAT. TXT, .CMD fera afficher la liste de tous les fichiers du disque de travail ayant comme extension TXT et CMD ; CAT B.TXT ferait afficher la liste des fichiers dont le nom commence par B et dont l'extension est TXT. La taille des fichiers est indiquée en nombre de secteurs ; nous vous rappelons à ce sujet qu'un secteur contient 256 octets dont 252 sont « utiles », les 4 autres étant pris par le DOS pour ses besoins propres. La protection des fichiers est indiquée par une ou deux lettres qui peuvent être un D ou un W selon que le fichier est protégé en effacement (D) ou en écriture (W). Il est également possible de protéger un fichier pour qu'il n'apparaisse pas au catalogue ; voyez à ce sujet la commande PROT.

CHIER2. L'extension pour FICHIER2, si elle n'est pas spécifiée, est identique à celle de FICHIER1. Pour chacun des fichiers, si un numéro de lecteur n'est pas précisé, celui de travail est pris par défaut. Si FICHIER2 existe déjà sur le disque (dans le cas de copie de disque à disque), l'autorisation de l'effacer vous est demandée et, en cas de réponse négative, la commande est terminée prématurément.

La deuxième forme de syntaxe permet de copier un fichier d'un disque sur un autre en lui conservant son nom original. Ici aussi, il est indispensable de préciser l'extension du nom du fichier ainsi copié. Si le numéro de lecteur du fichier à copier n'est pas précisé, celui-ci est pris sur le lecteur de travail ; ainsi COPY TOTO.TXT,0 fera copier le fichier TOTO.TXT pris sur le lecteur de travail sur le disque numéro 0.

La troisième forme de syntaxe est la plus puissante de toutes puisqu'elle permet de recopier un disque entier sur un autre ou seulement une portion qui est alors précisée par la liste de fichiers. Cette liste fonctionne comme dans la commande CAT vue ci-avant. Ainsi : COPY 0,1 recopiera tous les fichiers du disque 0 sur le disque 1, alors que COPY 0,1, .TXT, .CMD fera copier de 0 sur 1 tous les fichiers dont l'extension est TXT, puis tous les fichiers dont l'extension est CMD. Si des fichiers existent déjà sur le disque destination, l'autorisation de les effacer est demandée mais, si elle est refusée, la commande continue en séquence et n'est pas interrompue pour autant.

Nous vous incitons à essayer longuement cette commande afin de bien en maîtriser toutes les possibilités qui, lorsqu'on les connaît bien, font économiser pas mal de temps au niveau de la frappe de la commande elle-même.

Cette commande ne peut être utilisée si vous possédez un seul lecteur, et il vous faut dans ce cas faire appel à la commande ci-après, tout aussi puissante que celle-ci sinon plus. Cette deuxième forme de COPY n'est d'ailleurs pas limitée aux possesseurs d'un seul

```
+++TTYSET
BS = $0B
DL = $1B
EL = $3A
DP = 0
WD = 0
NL = 4
TB = $00
BE = $0B
EJ = 0
PS = ARRET
ES = $1B
+++TTYSET PS=0
+++TTYSET
BS = $0B
DL = $1B
EL = $3A
DP = 0
WD = 0
NL = 4
TB = $00
BE = $0B
EJ = 0
PS = MARCHÉ
ES = $1B
```

```
ASN W=1
DELETE TOTO.BIN
COPY 0,1, .BIN
```

Fig. 5. – Exemple de fichier utilisable par la commande EXEC.

```
+++VERIFY
VERIFICATION EN MARCHÉ
DESIREZ VOUS LA MODIFIER ? 0
VERIFICATION ARRETEE
+++VERIFY VM
VERIFICATION EN MARCHÉ
+++VERIFY
VERIFICATION EN MARCHÉ
DESIREZ VOUS LA MODIFIER ? N
+++
```

Fig. 6. – Exemple d'exécution de la commande VERIFY.

Fig. 4. – Exemple d'exécution de la commande TTYSET.

Commande COPY

Comme son nom l'indique, cette commande permet de copier des fichiers au sens large. Trois formes différentes de syntaxe sont possibles selon la fonction exacte désirée, ces formes sont :

– COPY <FICHIER1>, <FICHIER2>
– COPY <FICHIER>, <NUMERO>
– COPY <NUMERO>, <NUMERO> (,LISTE DE FICHIERS)

La première forme copie le fichier FICHIER1 dont le nom doit impérativement être spécifié avec son extension sous le nom FICHIER2. La copie peut avoir lieu de disque à disque ou sur le même disque ; dans ce dernier cas, FICHIER1, doit être un nom différent de FI-

lecteur et peut aussi être utilisée si vous disposez de plusieurs lecteurs.

Commande COPYSD

Cette commande permet de copier des fichiers d'une disquette sur une autre en utilisant un seul lecteur avec un maximum de souplesse. Le fonctionnement en est le suivant : la disquette comportant la commande COPYSD étant en place dans le lecteur, frappez COPYSD (LISTE DE FICHIERS) où LISTE DE FICHIERS répond aux mêmes règles que celles vues pour CAT ou COPY. Si vous ne donnez aucune liste de fichier, la commande vous posera les questions sur les fichiers que vous voulez recopier ou non. Vous pouvez aussi frapper une astérisque à la place de la liste de fichiers, auquel cas COPYSD considérera que vous souhaitez copier toute la disquette. La commande vous demande alors de mettre en place la première disquette (celle sur laquelle se trouvent le ou les fichiers à recopier) ; ceux-ci sont alors mis en mé-

moire jusqu'à ce qu'ils aient été tous lus ou que la mémoire soit pleine ; la commande vous demande alors de mettre en place la seconde disquette (celle devant recevoir les fichiers ainsi copiés) et y place les fichiers contenus en mémoire. Ce processus se répète alors jusqu'à complète exécution de la commande. Lorsque la mise en place d'une disquette vous est demandée, il faut frapper un caractère quelconque au clavier (sauf A) pour indiquer à COPYSD que c'est fait. Le fait de frapper A lors de la demande d'une disquette ou lors d'une demande d'auto-risation d'effacer un fichier déjà existant termine prématurément la commande.

Pour vous éviter des erreurs, cette commande vérifie, à chaque insertion de disquette, que vous avez bien mis en place la disquette attendue ; pour que cette vérification soit possible, il vous est interdit de copier ainsi deux disquettes l'une sur l'autre si elles ont le même nom, le même numéro de volume et la même date de création ; il faut au moins qu'un de ces trois paramètres soit différent entre les deux disquettes.

De nombreux messages peuvent apparaître pendant l'exécution de cette commande mais, comme ils sont en clair et en français, nous estimons qu'il est inutile de les décrire puisque leur signification vous sera évidente.

Précisons que si vous utilisez cette commande alors que vous avez plusieurs lecteurs, elle fonctionne exactement de la façon exposée ci-avant mais en considérant que le « seul » lecteur est celui de travail. Ce mode de fonctionnement est très utile si vous avez, par exemple, un lecteur 80 pistes et un lecteur 40 pistes, car c'est alors le seul moyen de faire des copies de 40 pistes à 40 pistes ou de 80 pistes à 80 pistes (les copies de 40 à 80 et de 80 à 40 sont, elles, toujours possibles grâce à la commande COPY puisque, comme expliqué au début de cet article, le DOS tient compte automatiquement du format du disque utilisé).

Commande DATE

Elle permet de changer la date fournie au système lors du chargement du DOS ou de

connaître celle-ci. Cette commande et cette notion de date ne sont pas à négliger comme le font à tort beaucoup de gens ; en effet, c'est une information qui est placée dans le répertoire des fichiers d'un disque et qui est prise en compte par plusieurs commandes telles que LIST, DIR (cette commande fait partie des extensions du DOS de base) et l'assembleur, entre autres. Elle permet très facilement de s'y retrouver, lorsque l'on commence à avoir un stock de disquettes important, par examen de sa valeur qui permet de savoir quelle est la version la plus récente d'un fichier donné. La syntaxe de la commande est la suivante :

— DATE (JJ,MM,AA) où JJ,MM, AA représente le numéro du jour, du mois, et les deux derniers chiffres de l'année. Si ces valeurs sont omises, la commande DATE seule fait imprimer la date courante. Si des valeurs incohérentes sont frappées (plus de 31 jours par mois ou plus de 12 mois par an), elles sont refusées. Cette vérification n'a pas lieu lors de l'entrée initiale de la date au chargement du DOS, et cette commande peut être utilisée pour corriger ainsi une éventuelle frappe fantaisiste.

Commande DELETE

Cette commande a pour fonction d'effacer un ou plusieurs fichiers sur une disquette. Son utilisation doit être faite avec attention car il est ensuite très difficile (mais possible) de récupérer un fichier ainsi effacé. Pour ce faire, la commande demande deux confirmations de l'ordre d'effacement ; c'est peut-être un peu fastidieux, mais cela présente l'avantage d'éviter quelques erreurs. La syntaxe est la suivante :

— DELETE <NOM DE FICHIER> (,NOM DE FICHIER,...) où NOM DE FICHIER doit être précisé avec son extension, le numéro de lecteur choisi étant celui de travail si vous ne le précisez pas. Plusieurs fichiers peuvent ainsi être spécifiés à la suite les uns

+++FORMAT 1

---- ATTENTION ! FORMATAGE ! ----

TOUT CE QUI EST SUR LA DISQUETTE A FORMATER VA ETRE DETRUIT

DISQUE VIERGE EN VOIE 1 ? 0
35 , 40 OU 80 PISTES (3 / 4 / 8) ? 4
DISQUE DOUBLE FACE ? 0
NOM DE VOLUME ? EXEMPLE
NUMERO DE VOLUME ? 1983

PATIENTEZ - FORMATAGE EN COURS

FORMATAGE TERMINE
SECTEURS DISPONIBLES = 780

+++

Fig. 7. - Exemple d'exécution de la commande FORMAT pour un disque 40 pistes double face.

+++DATE
29 DECEMBRE 1982
+++DATE 01,01,83
+++DATE
1 JANVIER 1983
+++DATE 01,13,83
Erreur de syntaxe commande - frapper à nouveau la commande
+++

Fig. 8. - Exemple d'exécution de la commande DATE avec une erreur (volontaire !).

des autres. Pour chacun, la commande va vous demander si vous voulez l'effacer et, si vous répondez O ou Y (pour oui ou yes), elle va vous demander si vous êtes bien sûr de vous ; une réponse négative laisse le fichier concerné intact.

Commande EXEC

Cette commande a pour fonction de faire exécuter automatiquement une suite de commandes du DOS définies au préalable dans un fichier. En d'autres termes, au moyen de la commande BUILD ou de l'éditeur, vous pouvez réaliser un fichier texte contenant une suite de commandes du DOS que vous souhaitez voir s'exécuter seules ou dont l'enchaînement revient souvent. Dès que vous souhaitez faire exécuter celles-ci, il vous suffira de frapper EXEC < NOM DE FICHIER > pour que les commandes contenues dans le fichier s'exécutent en séquence. Si l'extension n'est pas précisée pour le nom de fichier, elle est prise égale à TXT, et le disque choisi par défaut est celui de travail. Le fichier contenant les commandes doit être réalisé de la même façon que si vous frappiez les mêmes commandes clavier ; c'est-à-dire qu'il doit contenir une commande par ligne (ou plusieurs commandes sur la même ligne séparées par deux points, mais cela le rend moins lisible). La figure 5 donne un exemple de contenu de fichier pouvant être utilisé par la commande EXEC. Ce fichier aurait pour effet d'affecter le lecteur numéro 1 comme disque de travail, d'effacer le fichier TOTO.BIN sur le disque 1 et de copier, du disque 0 au disque 1, tous les fichiers avec l'extension BIN.

Commande FORMAT

Cette commande a pour effet de formater une disquette, c'est-à-dire de préparer celle-ci pour qu'elle puisse être ensuite utilisée par le DOS. Cette préparation consiste à y écrire un certain

nombre d'informations auxquelles vous n'aurez pas accès mais qui sont indispensables au DOS. De plus, cette commande teste tous les secteurs d'une disquette et enlève automatiquement de la table des secteurs disponibles ceux qui sont défectueux, s'il y en a. La syntaxe est la suivante :

– FORMAT <NUMERO> où NUMERO est le numéro du lecteur contenant la disquette à formater. Attention ! comme c'est rappelé par un message en début de formatage, tout ce qui se trouvait sur une disquette avant formatage est irrémédiablement détruit par l'opération de formatage, sans possibilité de récupération ultérieure. La commande FORMAT vous pose un certain nombre de questions pour savoir si vous voulez faire une disquette 35, 40 ou 80 pistes ; répondez en fonction de vos lecteurs (le 35 pistes a été prévu pour ceux qui possèdent d'anciens lecteurs car, maintenant, quasiment tous les modèles sont 40 pistes) ; elle vous demande si vous voulez une disquette simple ou double face ; répondez, ici encore, en fonction de vos lecteurs, puis un nom (dit « nom de volume ») vous est demandé ; ce nom sera celui de la disquette, et il peut contenir jusqu'à huit lettres, chiffres ou tiret et ne doit pas commencer par un chiffre (même contrainte que pour les noms de fichiers) ; enfin, un numéro vous est demandé, numéro qui peut comporter jusqu'à quatre chiffres et qui sera le numéro de la disquette. Le formatage commence alors. C'est une opération assez longue, surtout pour une disquette 80 pistes double face en raison du test de chaque secteur qui y est réalisé. Les secteurs défectueux sont indiqués sur l'écran sous la forme XXYX où XX est le numéro de piste et YY le numéro de secteur. En fin de formatage, le nombre total de secteurs disponibles est indiqué. Certaines zones de la disquette étant indispensables au DOS (au niveau de la piste 0), un mauvais secteur dans celles-ci rend la disquette inutilisable ; un message est alors affiché pour vous en avertir. Le message

FORMATAGE INTERROMPU peut de plus apparaître si vous essayez de formater une disquette protégée en écriture, ou si vous avez mal inséré votre disquette dans le lecteur et si « elle tourne carré » (voir en fin d'article). Si vous n'avez qu'un lecteur, il vous faut, pour formater, procéder de la sorte :

– Insérez la disquette DOS supportant la commande FORMAT dans le lecteur.
– Frappez FORMAT ou FORMAT O.
– Lorsque le message de la commande FORMAT apparaît, enlevez la disquette DOS et placez alors la disquette à formater dans le lecteur.
– Procédez ensuite comme indiqué ci-avant, pour ceux qui possèdent deux lecteurs.

Commande GET

Cette commande fait partie des commandes résidentes du DOS et est donc chargée en mémoire en permanence ; la présence de la disquette DOS n'est donc pas nécessaire pour pouvoir l'exécuter. Sa fonction est de charger un fichier en mémoire. La syntaxe est fort simple :

– GET <NOM DE FICHIER> (, <NOM DE FICHIER>, ...) où NOM DE FICHIER est le nom du fichier à charger en mémoire. Ce nom peut être seul ou suivi d'autant d'autres noms que vous le souhaitez ; la commande chargera alors tous ces fichiers en mémoire, les uns après les autres. L'extension prise par défaut est BIN et le disque par défaut celui de travail. L'adresse de chargement en mémoire est celle contenue dans le fichier lui-même et n'a pas à être précisée. Si vous souhaitez charger un fichier en mémoire ailleurs qu'à sa place normale, il faut utiliser la commande OLOAD des extensions du DOS.

Commande I

Cette commande complète EXEC bien qu'elle puisse être également utilisée seule, dans certains cas. Elle a pour fonction de remplacer les informa-

tions que vous devez frapper au clavier en réponse à certaines commandes par celles contenues dans le fichier spécifié. La syntaxe générale de cette commande est la suivante :

– I <NOM DE FICHIER>, <COMMANDE> où NOM DE FICHIER est le nom d'un fichier contenant les réponses aux questions qui seront posées par la commande qui suit. L'extension prise par défaut est TXT et le disque par défaut est celui de travail. Ainsi, si vous voulez rendre l'effacement d'un fichier automatique, c'est-à-dire ne pas avoir à répondre deux fois de suite oui aux questions posées lors d'un DELETE, vous pourrez réaliser, avec BUILD ou avec l'éditeur de texte, le fichier que nous allons appeler OUI.TXT qui contiendra OO (deux lettres O à la suite). Lorsque vous frapperez ensuite I OUI DELETE TOTO.BIN, la commande DELETE ira chercher les deux réponses aux questions qu'elle pose dans OUI (.TXT pris par défaut) et, comme elle y trouvera deux oui, elle effacera le fichier TOTO.BIN.

Il est évident que l'intérêt de cette commande est très important dans des fichiers pour la commande EXEC, car cela permet ainsi d'enchaîner dans un EXEC des commandes qui nécessitent des réponses à des questions.

Commande JUMP

Cette commande permet de sauter directement à une adresse mémoire et de commencer l'exécution du programme qui est sensé s'y trouver. La syntaxe en est fort simple :

– JUMP <ADRESSE> où ADRESSE est constituée de 1 à 4 chiffres hexadécimaux représentant l'adresse où sauter. Attention ! cette commande est la seule qui puisse « planter » le DOS. En effet, celui-ci ne dispose d'aucun moyen de vérifier si vous avez bel et bien un programme présent à l'adresse ainsi spécifiée et, si vous n'avez rien ou si le programme qui se trouve à cette adresse n'est pas au point, il peut se passer n'importe quoi.

Si l'adresse donnée lors de cette commande comporte des lettres comprises entre A et F (puisque l'adresse est en hexadécimal), celles-ci doivent être frappées en majuscules.

Commande LINK

Cette commande n'est utilisée que dans un cas : lorsque vous réalisez des disquettes contenant le DOS que vous souhaitez ensuite pouvoir utiliser comme disquettes système. En effet, lorsque vous frappez la commande X sous TAVBUG09, un programme contenu dans TAVBUG09 va chercher sur la disquette un autre programme appelé le chargeur du DOS. Ce chargeur est toujours stocké au même endroit sur la disquette, et il n'y a donc pas de problème pour le trouver. Ce chargeur va, à son tour, aller chercher le DOS pour le mettre en mémoire, mais le DOS peut se trouver n'importe où sur la disquette ; il faut donc informer ce chargeur de l'emplacement du DOS sur celle-ci, et cela se fait au moyen de la commande LINK. Ainsi, pratiquement, lorsque vous fabriquez une disquette devant contenir le DOS, vous allez faire un FORMAT puis copier sur la disquette ainsi formatée le DOS et tous les fichiers qui vous intéressent ; puis vous ferez un LINK qui indiquera au chargeur où se trouve le DOS (le chargeur n'a pas à être copié sur la disquette, c'est fait automatiquement lors de la commande FORMAT). La syntaxe est très simple :

– LINK TAVDOS09.SYS, puisque vous n'avez pas le choix du nom de fichier. Le disque sur lequel agit le LINK est celui de travail si cela n'a pas été précisé.

Commande LIST

Cette commande a pour fonction de faire lister sur le terminal ou sur une imprimante le contenu d'un fichier. La syntaxe en est la suivante :

– LIST <NOM DE FICHER> (<LIGNE>) (<+OPTION>) où NOM DE

FICHER est le nom du fichier à lister ; l'extension par défaut étant TXT et le lecteur par défaut étant celui de travail. Les informations qui suivent sont facultatives et correspondent à des extensions de la commande. Ainsi, LIGNE permet de préciser à partir de quelle ligne et jusqu'à quelle ligne le fichier sera listé. LIGNE peut avoir la forme : NN ou NN-MM, où NN représente le numéro de la ligne où commencera le listing et MM le numéro de ligne où il finira. Si NN seul est précisé, le fichier sera listé de NN jusqu'à la fin. Des options supplémentaires sont possibles et sont matérialisées par une ou deux lettres placées après le signe plus (+). L'option N fait imprimer en début de chaque ligne listée son numéro, tandis que l'option P fait paginer le listing en imprimant en haut de chaque page un titre (qui vous est demandé par cette commande lorsque vous avez choisi l'option P) qui peut comporter jusqu'à 40 caractères, la date courante du système et un numéro de page. Cette pagination est réelle et prend sa signification si vous travaillez avec une imprimante, puisque chaque page est suivie par le caractère OC qui est le FORM FEED reconnu par toutes les imprimantes dignes de ce nom.

Voici quelques exemples de cette commande : LIST O.ERREURS.SYS fait lister sur le terminal le contenu du fichier ERREURS.SYS (liste des messages d'erreur du DOS) ; LIST TOTO,100,+NP fait lister le fichier TOTO.TXT pris sur le lecteur de travail de la ligne 100 à la fin en paginant le listing et en imprimant les numéros de lignes.

Commande MON

Cette commande est une commande résidente du DOS et, de ce fait, elle est toujours présente en mémoire, que le disque DOS soit en place ou non. Sa fonction est de permettre de passer sous le contrôle de TAVBUG09 de façon « propre ». La syntaxe est tout simplement :

– MON

Lorsque vous souhaitez revenir sous le DOS, il vous est

possible de faire appel à la commande Y de TAVBUG09. Si vous avez détruit le DOS en mémoire, il vous faut alors faire une commande X pour charger à nouveau celui-ci.

Commande O

Cette commande est un peu la contrepartie de la commande I, en ce sens qu'elle dirige tout ce qui serait affiché sur le terminal, suite à la frappe d'une commande, dans un fichier. Sa syntaxe en est la suivante :

– O <NOM DE FICHER>, <COMMANDE> où NOM DE FICHER est le nom d'un fichier dans lequel vont être placées toutes les informations que la commande qui suit aurait affichées sur l'écran du terminal. Le fichier est placé par défaut sur le lecteur de travail et son extension par défaut est OUT (et non TXT, pour rappeler que c'est un fichier provenant de la commande O).

Cette commande est surtout utile lorsque l'on fait assembler un programme et que l'on souhaite en garder un listing sur disque pour une utilisation ultérieure (impression du listing sur le système d'un ami qui possède une imprimante, par exemple). Ainsi : O LISTING ASMB TOTO.TXT fera assembler (commande ASMB non encore étudiée) le programme TOTO.TXT et placera le listing résultant dans le fichier LISTING.OUT sur le disque de travail.

Commande P

Cette commande n'en est en fait pas une puisqu'elle peut se combiner avec n'importe laquelle des commandes DOS décrites ici ou à venir par la suite. Cette « commande » permet de faire fonctionner une imprimante, en ce sens qu'elle dirige l'impression des informations utiles (essayer un P LIST <NOM DE FICHER>,+P pour comprendre la signification de « utiles » sur une imprimante. Elle s'utilise de la façon suivante :

– P <COMMANDE> où COMMANDE est N'IMPORTE QUELLE COMMANDE COMPATIBLE DU DOS !

Tel que le DOS vous est fourni, il est préparé pour fonctionner avec une imprimante quelconque connectée sur la sortie imprimante de la carte CPU09, c'est-à-dire qu'il est prévu pour une imprimante disposant d'une interface CENTRONICS (voir nos précédents articles à ce sujet). Si vous utilisez la carte IVG09 et non le terminal vidéo, il vous est possible de récupérer la sortie série RS232 ainsi libérée sur la carte CPU09 pour y connecter une imprimante à interface série (telle que la Heathkit H 14 de notre ancien système, par exemple). Nous indiquons dans la suite de cet article ce qu'il convient de faire pour que cela fonctionne.

Attention, si vous frappez une commande P alors qu'aucune imprimante n'est présente ou alors que l'imprimante n'est pas « On line », le système va se bloquer en attente de la disponibilité de celle-ci. Vous ne pourrez alors plus sortir de cet état qu'en mettant réellement une imprimante en service ou en faisant un RESET suivi par un Y pour ré-entrer sous le DOS.

Commande PROT

Cette commande a pour fonction de protéger un fichier ou au contraire d'enlever des protections sur un fichier. La syntaxe en est la suivante :

– PROT <NOM DE FICHER>, <LETTRE> où NOM DE FICHER représente le nom du fichier sur lequel doit agir PROT. L'extension doit être précisée, le lecteur choisi par défaut est celui de travail. LETTRE représente une ou plusieurs des lettres ci-après et correspond à la fonction désirée :

– W protège un fichier en écriture ; il est alors impossible de l'effacer, de le changer de nom et d'y écrire quoi que ce soit ;

– D protège un fichier en effacement ; il est alors impossible de l'effacer, ou de le changer de nom, mais on peut très bien y écrire pour y ajouter des informations ;

– C protège un fichier vis-à-vis du catalogue. Un tel fichier fonctionne comme par le

passé mais n'apparaît plus lors d'une commande CAT ;

- X enlève toutes les protections sur un fichier.

Ces lettres peuvent être combinées ; ainsi : PROT TOTO.BIN,XW enlève toutes les protections qui pouvaient exister sur TOTO.BIN puis protège ensuite celui-ci en écriture.

Commande RENAME

Cette commande a pour fonction de changer le nom d'un fichier. La syntaxe en est la suivante :

- RENAME <NOM DE FICHER>, <NOUVEAU NOM> où NOM DE FICHER est le nom du fichier à changer et où NOUVEAU NOM est le nouveau nom qui va être donné au fichier. Si l'extension n'est pas spécifiée pour le nom initial, TXT est prise par défaut et le fichier est pris sur le lecteur de travail. Il est inutile de préciser le lecteur pour le nouveau nom puisque celui-ci est obligatoirement le même que pour l'ancien nom. De plus, si aucune extension n'est précisée pour le nouveau nom, c'est celle de l'ancien qui est prise par défaut.

Cette commande est celle que vous devez utiliser si, comme nous l'avons expliqué en début d'article, vous souhaitez mettre aussi en français les commandes du DOS ; ainsi pourrez-vous faire par exemple :

- RENAME O.DELETE.COMD,EFFACE pour donner à la commande DELETE l'appellation plus française EFFACE.

Commande SAVE

Cette commande a pour fonction de sauvegarder sur disque une portion quelconque de mémoire. La syntaxe en est la suivante :

- SAVE <NOM DE FICHER>, <DEBUT>, <FIN> (,<TRANSFERT>) où NOM DE FICHER est le nom qui va être donné au fichier contenant ce que vous allez ainsi sauvegarder. L'extension prise par défaut est BIN et le lecteur par défaut est

celui de travail. DEBUT et FIN sont respectivement les adresses de début et de fin de la zone mémoire à sauvegarder, exprimées en hexadécimal. Si ces adresses comprennent des lettres de A à F, elles doivent impérativement être frappées en majuscules. TRANSFERT est ce que l'on appelle sous ce DOS une adresse de transfert, c'est-à-dire une adresse de début d'exécution de programme lorsque le fichier ainsi constitué est ensuite considéré comme une commande du DOS. Cette notion sera précisée dans l'étude approfondie du DOS proposée dans la suite de cet article. Si le nom de fichier spécifié existe déjà sur le disque, l'autorisation de l'effacer est demandée, et le fait de répondre non termine prématurément la commande.

Si vous souhaitez sauvegarder des morceaux de mémoire discontinus en un seul fichier, il est préférable de les sauvegarder un par un et de faire ensuite un APPEND des fichiers ainsi créés ; la place occupée sur le disque sera moindre que si vous sauvegardiez un bloc de mémoire englobant tous les morceaux désirés.

Comme toutes les commandes transitoires du DOS, celle-ci réside en mémoire à partir de C100, et en conséquence, si vous voulez sauvegarder une portion de mémoire située dans cette zone, il y a impossibilité. Il faut, dans ce cas, faire appel à la commande SAVE.LOW qui s'utilise exactement comme SAVE mais qui, elle, réside de 0100 à 02A0. Ces notions seront, elles aussi, précisées dans l'étude approfondie du DOS.

Commande TTYSET

Cette commande a pour fonction de définir certains paramètres relatifs au terminal utilisé, aux caractères de contrôle et à leur signification, ainsi que diverses autres fonction décrites ci-après. La syntaxe en est la suivante :

- TTYSET (LISTE DE PARAMETRES). Si la liste de paramètres n'est pas précisée, le fait de frapper TTYSET seul entraîne l'affichage de la valeur actuellement affectée aux

divers paramètres. Cette liste est à spécifier au moyen d'une succession de deux lettres auxquelles on affecte une valeur par un signal égal (XX=AA, YY=BB, ZZ=CC, etc.) ; ces lettres et les valeurs que l'on peut leur donner étant précisées ci-après.

- BS sert à définir le code du caractère de retour arrière du curseur. Il est mis à 08 (code du CNTRL H ou de la flèche vers la gauche) lors de l'initialisation du DOS et peut être remplacé par n'importe quel code. Le fait de lui donner comme valeur 00 annule cette fonction de retour arrière.

- BE sert à définir le caractère que le DOS envoie au terminal lorsqu'il a reçu un CNTRL H ; en effet, le CNTRL H signifie retour arrière du curseur pour la majorité des terminaux, mais pour le DOS il signifie aussi effacement du caractère sur lequel le curseur est revenu, ce que ne comprennent pas tous les terminaux. Ce paramètre BE permet donc d'envoyer au terminal ce que vous désirez. De plus, si vous mettez BE=08, le DOS fera en sorte que l'écran de votre terminal se comporte comme le DOS, c'est-à-dire que vous verrez bel et bien s'effacer le caractère sur lequel sera revenu le curseur.

- DL sert à définir le code du caractère de contrôle qui réalise l'effacement d'une ligne. On peut lui donner n'importe quelle valeur ; à l'initialisation du DOS, DL=18, qui est le code ASCII du CNTRL X.

Nous ne vous conseillons pas de changer ces trois premiers paramètres, car ils sont standardisés au niveau de l'immense majorité des terminaux du marché, et il est ridicule de se mettre volontairement hors standard soi-même, surtout pour de telles choses.

- EL sert à définir le caractère de fin de ligne, c'est-à-dire celui qui est utilisé pour séparer plusieurs commandes sur la même ligne. On peut lui donner n'importe quelle valeur et le fait de lui donner la valeur 00 annule cette fonction ; à l'initialisation du DOS, ES=3A, qui est le code ASCII des deux points (:).

- DP sert à préciser au DOS qu'une page est constituée de

NN lignes sous la forme suivante, DP=NN où NN est un nombre décimal. Le fait de mettre DP=00 annule cette possibilité, ce qui est le cas à l'initialisation du DOS. Cette fonction est indépendante de l'option +P de la commande LIST qui, elle, fonctionne toujours pour du papier standard au pliage accordéon, quelle que soit la valeur de DP.

- WD sert à indiquer au DOS le nombre de caractères qu'il doit afficher par ligne, sous la forme WD=NN où NN représente ce nombre exprimé en décimal. Si une ligne plus longue que WD doit être affichée, le DOS la coupera automatiquement en morceaux à chaque multiple de WD ; ainsi, si WD=60 et qu'il faille afficher une ligne de 130 caractères, les 60 premiers seront affichés sur une ligne, les 60 suivants sur une autre et les 10 derniers sur une troisième. Si WD=00, cette fonction est annulée, ce qui est le cas à l'initialisation du DOS.

- NL sert à indiquer au DOS combien il doit envoyer de caractères nuls lors de chaque retour chariot. Cette fonction est prévue pour ceux d'entre vous qui utilisent un terminal autre que le terminal vidéo de décembre 1981 ou les cartes IVG et IVG09 et qui peuvent avoir parfois besoin de cette fonction. Ce paramètre est mis à 00 à l'initialisation du DOS.

- TB permet de définir le caractère de tabulation ; il n'est pas utilisé par le DOS mais d'autres logiciels, tel l'éditeur, y font appel. L'on peut donner à TB n'importe quelle valeur et, à l'initialisation, il est mis à 00, ce qui désactive cette fonction.

- EJ indique au DOS combien il doit faire sortir de sauts lignes en fin de chaque page définie au moyen de la commande DP. Cela permet de sauter par-dessus les perforations du papier à pliage accordéon. A l'initialisation, EJ est mise à 00, ce qui dévalide cette fonction.

- PS permet d'indiquer au DOS qu'il doit ou non faire une pause en fin de chaque page définie par DP. Si PS=Y ou PS=0, le DOS s'arrêtera à la fin de chaque page et attendra que vous frappiez le caractère

ESCAPE (ESC sur certains claviers). Si PS=N (pour non), cette fonction est inactive. Si DP a été laissée à 00, même si vous faites PS=O ou Y (pour Oui ou Yes), la pause ne fonctionnera pas puisque vous n'aurez pas défini de taille de page.

— ES sert à définir le code affecté au caractère ESCAPE dont nous avons déjà parlé plusieurs fois. N'importe quelle valeur peut être donnée à ce caractère mais, à l'initialisation, ES=1B, qui est le vrai code ASCII de ESCAPE.

Cette commande TTYSET appelle plusieurs commentaires : le premier est que nous vous déconseillons de l'utiliser pour modifier un paramètre tant que vous n'avez pas bien pris l'habitude d'utiliser le DOS. En effet, celui-ci vous est fourni avec les valeurs les plus logiques qu'il soit pour tous ces paramètres et, en principe (sauf si vous avez vraiment un terminal spécial), vous n'avez rien à modifier. Le deuxième est que la fonction des caractères de définition des formats de pages n'est pas prise en compte par tous les logiciels tournant avec ce DOS (LIST avec l'option P est un exemple, mais il y en a d'autres). En conséquence, il est, et de loin, préférable d'utiliser du papier plié en accordéon de taille standard sur votre imprimante plutôt que de faire appel à ces fonctions.

Remarquez que lorsque vous faites un TTYSET seul, les paramètres sont affichés avec les conventions classiques ; à savoir que toutes les valeurs qui sont en hexadécimal sont précédées du symbole dollar. Par contre, vous n'avez pas à frapper ce symbole lorsque vous définissez une valeur, le DOS sachant très bien ceux des paramètres de TTYSET qui sont en hexadécimal et ceux qui ne le sont pas.

Commande VERIFY

Cette commande a pour fonction de mettre en marche ou d'arrêter la vérification systématique que fait le DOS après chaque écriture sur disque. Si la vérification est en

marque, toutes les fois que le DOS est amené à écrire sur un disque, il relit ce qu'il a écrit immédiatement après et le compare à ce qu'il aurait dû écrire pour voir si c'est identique ; cette pratique confère donc une grande sûreté de fonctionnement au système ; en contrepartie, elle ralentit le DOS dans un rapport pouvant aller jusqu'à deux, et c'est pour cela que nous avons prévu la possibilité de l'arrêter, bien que nous estimions cela dangereux. La syntaxe est la suivante :

— VERIFY ou VERIFY VM ou VERIFY VA. Dans le premier cas, l'état du DOS est affiché et la commande demande si vous voulez le modifier (répondez par O ou Y pour oui, ou N pour non). Dans le deuxième cas, la vérification est mise en marche ; dans le troisième cas, elle est arrêtée (VM = Vérification Marche et VA = Vérification Arrêt).

Commande XOUT

Cette commande a pour effet d'effacer tous les fichiers munis de l'extension OUT contenus sur une disquette (autres que ceux qui sont protégés). Rappelons que les fichiers munis de cette extension sont ceux qui ont été produits par la commande O vue ci-avant. La syntaxe est la suivante :

— XOUT (NUMERO) où NUMERO est le numéro du lecteur sur lequel doit agir la commande. S'il n'est pas précisé, le lecteur de travail est utilisé.

L'éditeur l'assembleur et le reste

Ce sont évidemment les commandes les plus puissantes et les plus intéressantes de ce DOS de base. Il est tout aussi évident que, vu la place que nous avons déjà occupée dans ce numéro et vu la quantité d'informations qu'il nous faut vous fournir à leur sujet, cela ne sera possible que dans le prochain article de cette série. Pour ce qui est de l'éditeur, cela ira vite puisque 80 % des commandes de l'éditeur cassette s'y retrouvent. Pour l'assembleur, ce sera un peu plus long, mais

nous vous garantissons un passage en un seul numéro.

Ce DOS de base comporte de nombreuses commandes, mais il est encore plus souple et puissant d'emploi avec les extensions dont nous parlons dans les informations 6809 ; ces extensions ne sont autres que 26 commandes supplémentaires qui seront décrites après l'éditeur et l'assembleur. Ces commandes parlent, comme le DOS de base, l'éditeur et l'assembleur, en français.

Le BASIC, qui est un modèle très performant, sera disponible à la fin de ce mois et il présentera la particularité intéressante d'indiquer les erreurs en français et en clair, plutôt que par le trop classique ERREUR ZZ LIGNE 26.

Les autres logiciels vous seront annoncés au fur et à mesure de leur mise à disposition ; de plus, nous allons faire le nécessaire pour qu'une société spécialisée importe, principalement des USA, certains logiciels qui pourront aussi tourner sur ce système (et ils sont de plus en plus nombreux vu le « boum » du 6809 chez les amateurs américains).

Enfin, et pour en terminer avec ces informations logicielles, nous décrivons dès que possible la face cachée du DOS, c'est-à-dire les sous-programmes que vous pouvez utiliser, les constantes particulières, les possibilités de créer vos propres commandes, etc., de façon que vous sachiez tout à son sujet. Cette description sera même assortie des informations relatives au DOS 6800 puisque bien des points sont communs entre les deux.

Informations diverses

Les informations 6809 ont été rééditées à Noël et, en conséquence, la version la plus à jour de celles-ci aujourd'hui est datée du 20-12-82. Si vous ne la possédez pas encore, nous rappelons pour la nième fois que vous pouvez la demander à l'auteur en adressant au service du courrier de la revue et à son attention une enveloppe format 16 X 22 minimum affranchie à 9,60 F et libellée à votre adresse, ac-

compagnée d'un quelconque papier mentionnant « Informations 6809 ». Ne joignez aucune question à cette demande, car ce n'est pas toujours l'auteur qui s'en occupe et il ne pourrait donc vous être répondu. Les demandes non conformes aux indications ci-avant sont mises à la poubelle par souci d'équité ; en effet 99 % d'entre vous se conforment à ces indications, nous ne voyons donc pas pourquoi le 1 % restant bénéficierait de passe droit ! Pour nos amis lecteurs étrangers, l'affranchissement à 9,60 F est à remplacer par quatre coupons réponse internationaux (en vente dans tous les bureaux de poste) ; ne collez pas de timbres de votre pays qui seraient perdus pour vous et pour l'auteur puisque les postes françaises ne les acceptent pas.

Côté matériel, nous n'abandonnons pas pour autant et, dès ces notices de logiciel passées, nous allons reprendre les études de cartes avec, en particulier, une nouveauté qui nous a été demandée par ceux d'entre vous qui ne veulent pas ou ne peuvent pas passer à la version disquettes. Ce sera une carte UVPRM très moderne (puisque supportant tout de la 2716 à la 2764), que l'on pourra placer en page 0, et sur laquelle il sera possible de placer n'importe lequel de nos logiciels sur cassette, celui-ci étant alors chargé en page 1 dès que vous voulez l'utiliser, et ce en quelques millisecondes. Cela ne vaut pas une disquette, bien sûr, mais il est plus agréable d'avoir accès au BASIC ou à l'éditeur en quelques millisecondes plutôt qu'après plusieurs minutes d'attente, comme c'est le cas avec les cassettes.

Conclusion

Nous en restons là pour aujourd'hui, compte tenu de ce que nous avons précisé ci-avant relativement au logiciel et aux articles futurs, et vous souhaitons une bonne prise de contact avec ce système d'exploitation puisque c'est là le nom « officiel » en français du DOS.

C. TAVERNIER
(A suivre.)

Réalisez votre ordinateur individuel

Éditeur et macro assembleur sur disquette

COMME nous vous l'avons annoncé le mois dernier, nous allons vous présenter aujourd'hui le mode d'emploi de l'éditeur et du macro-assembleur sur disquette, qui sont, rappelons-le, fournis d'origine avec la disquette DOS.

L'éditeur sur disquette

La description de son mode d'emploi va être relativement rapide puisque ce programme est un sur-ensemble de l'éditeur sur cassette dont nous vous avons présenté la notice dans le numéro 1686 de novembre 1982. Nous allons donc vous demander de prendre ce numéro à côté de vous pour lire les lignes qui vont suivre et que nous allons présenter en conséquence.

Tout d'abord, nous allons voir les différences, très peu nombreuses, entre l'éditeur disque et l'éditeur cassette, puis nous verrons les commandes supplémentaires dont dispose celui-ci.

L'appel de l'éditeur

Pour lancer l'éditeur sur disquette, il suffit de frapper EDIT NOM DE FICHIER, où NOM DE FICHIER représente le nom que vous voulez donner au fichier que vous allez ainsi créer ou le nom d'un fichier déjà existant mais que vous désirez éditer. Si aucun numéro de lecteur n'est précisé, celui de travail est pris par défaut, et si aucune extension n'est précisée, TXT est prise par défaut ; ainsi, EDIT TOTO éditera le fichier TOTO.TXT sur le lecteur

de travail. Il est également possible d'utiliser la syntaxe suivante : EDIT FICHIER1, FICHIER2 ; dans ce cas, le fichier appelé FICHIER1 doit exister et va être chargé en mémoire de l'éditeur ; lorsque la session d'édition sera finie, le fichier ainsi édité sera sauvegardé sur disque sous le nom FICHIER2 (FICHIER1 restant, bien sûr, présent sur le disque tel qu'il était avant édition). Cette syntaxe respecte les règles ci-avant pour les numéros de lecteurs et les extensions.

Après la frappe de la commande sous une des formes ci-avant, l'éditeur se charge en mémoire, se lance, et deux cas peuvent alors se produire :

- Le nom de fichier spécifié n'existe pas ; dans ce cas, l'éditeur affiche le message NOUVEAU FICHIER et se place en mode d'insertion de ligne à partir de la ligne 1.00. Lorsque la session d'édition sera terminée, l'éditeur sauvegardera ce que vous aurez ainsi édité dans un fichier qu'il créera avec le nom que vous avez donné.
- Le nom de fichier spécifié existe ; dans ce cas, l'éditeur change son extension de ce qu'elle était (TXT en principe) en BAK et laisse le fichier ainsi débaptisé tel quel sur le disque ; de plus, il charge ce fichier dans sa mémoire de travail et se place ensuite en mode d'attente de commande. Lorsque la session d'édition

sera terminée, l'éditeur sauvegardera le fichier ainsi édité sur disque avec le nom spécifié et l'extension spécifiée (TXT par défaut). Cela signifie que vous aurez alors sur le disque deux fichiers : l'un muni d'une extension BAK qui est votre fichier avant édition, l'autre, de même nom mais avec extension TXT, qui est votre fichier après édition. Cette façon de faire est remarquablement intéressante, car elle permet, en cas de grosses erreurs lors de l'édition d'un fichier, de toujours avoir sous la main la version précédente pour pouvoir se rattraper.

Ce deuxième mode de fonctionnement conduit à plusieurs remarques dictées par le bon sens et par le vieil axiome « pourquoi faire compliqué lorsque l'on peut faire simple ? » :

- Il est fortement recommandé d'utiliser pour tous les fichiers à éditer l'extension par défaut TXT.
- Il est fortement déconseillé de donner à un fichier une extension BAK, car on risque alors de le confondre avec un fichier créé par l'éditeur.
- Il est évident que si l'on édite plus de deux fois un même fichier, il va y avoir un problème puisque, lors de la deuxième édition, l'éditeur va chercher à créer un nouveau fichier avec extension BAK, alors qu'il existe déjà puisqu'il avait été créé lors de l'édition précédente. L'éditeur va alors vous demander s'il peut effacer le fichier muni de l'extension BAK de l'édition précédente, le fait de répondre N (pour non) termine prématurément

l'édition. Ce fonctionnement est logique puisqu'il permet de conserver toujours une copie du fichier avant la dernière édition en cours, alors qu'il est parfaitement inutile de garder une copie d'un fichier à toutes les étapes d'édérations intermédiaires.

Pour terminer une édition

Bien que ce soit possible, comme nous allons le voir ci-après, au niveau des commandes supplémentaires par rapport à la version cassette, il n'est pas nécessaire de faire de SAVE ou de WRITE ; en effet, et comme expliqué ci-avant, lors de la fin d'une session d'édition, matérialisée par la commande S(TOP) ou LOG (même commande que pour l'éditeur sur cassette), l'éditeur disque sauvegarde automatiquement le fichier en cours d'édition sur le disque conformément aux indications de nom qui avaient été données lors de l'appel de l'éditeur. Le contrôle est ensuite rendu au DOS, ce qui est matérialisé par l'apparition des trois signes + caractéristiques. Attention, si le fichier édité est long, et malgré la rapidité des disquettes, il faudra un certain temps après la frappe de S ou LOG pour que le DOS reprenne la main.

Une remarque s'impose aussi, il est impossible d'effectuer une édition sur un disque protégé en écriture et, si vous le demandez, un message d'erreur vous rappellera gentiment à l'ordre.

Les caractères de contrôle

Leurs rôles sont identiques à ceux que nous avons définis pour l'éditeur sur cassette. Il faut cependant faire attention au fait que la commande TTYSET, vue le mois dernier, permet de redéfinir certains d'entre eux. Si vous utilisez le DOS tel qu'il vous est fourni, les définitions des caractères de contrôle de l'éditeur DOS sont identiques à celles de l'éditeur sur cassette ; si vous avez modifié la définition de certains d'entre eux, et en particulier de : EL, BS et DL, ces modifications seront prises en compte par l'éditeur disque qui se conformera alors à celles-ci. Ainsi, par exemple, le caractère d'effacement d'une ligne est CNTRL X sur l'éditeur cassette, le DOS d'origine et, donc, l'éditeur disque ; si, au moyen de TTYSET, vous avez décidé que ce caractère serait CNTRL Z ; ce sera CNTRL Z qui deviendra le caractère d'effacement de ligne pour l'éditeur disque. Cela nous conduit à renouveler le conseil que nous donnions le mois dernier au sujet de TTYSET : sauf besoin vraiment impérieux (terminal non standard, par exemple), ne changez pas ces caractères de contrôle.

Les nouvelles commandes

Compte tenu des possibilités très étendues de la version cassette de cet éditeur, elles sont très peu nombreuses, comme vous allez le constater ci-après.

Tout d'abord, et indépendamment de ce que nous avons exposé ci-avant sur l'appel de l'éditeur et la fin d'une édition, il est possible d'utiliser les commandes SAVE, READ et WRITE vues pour la version cassette, et ces commandes peuvent travailler avec les disquettes, avec les cassettes ou avec tout autre dispositif à votre convenance, comme indiqué ci-après au paragraphe adaptation à vos besoins. La commande GAP est toujours présente également mais n'a de signification qu'avec les

cassettes. Ces trois commandes (READ, WRITE et SAVE) fonctionnent comme expliqué pour la version cassette, mais, après la frappe de l'une d'elles, l'éditeur vous pose la question CASSETTE OU DISQUETTE ? à laquelle vous devez répondre par C ou D. Si vous répondez C, vous êtes ramené au cas de l'éditeur sur cassette ; si vous répondez D, l'éditeur vous demande le nom du fichier qui doit être utilisé par cette commande. En d'autres termes, cela signifie que, lors de l'édition d'un fichier dont le nom a été défini lors de la frappe de la commande EDIT, vous pouvez sauvegarder tout ou partie de celui-ci sous un autre nom de fichier (commandes SAVE et WRITE), mais vous pouvez aussi inclure dans votre fichier un tout autre fichier (commande READ). Cela contribue à accroître la souplesse de cet éditeur.

Tel que l'éditeur est fourni, le fait de répondre C lors des commandes SAVE, READ ou WRITE, lance les sous-programmes de dialogue, avec un magnétophone à cassette, contenus dans TAVBUG09 ; il est possible de modifier cela, comme indiqué ci-après, et de faire lancer les programmes de votre choix, ce qui permet, entre autres possibilités, de faire dialoguer cet éditeur avec quasiment n'importe quoi et, pourquoi pas, avec un autre micro-ordinateur (un ZX 81 dont vous voulez récupérer certains programmes Basic par exemple).

Les commandes vraiment nouvelles par rapport à la version cassette sont au nombre de trois, et nous allons en voir le rôle.

— La commande ABORT permet de terminer la session d'édition en cours et de rendre le contrôle au DOS, mais d'une façon très particulière. En effet, lorsque vous frappez ABORT, le fichier en cours d'édition n'est pas sauvegardé sur disque et son prédécesseur, qui s'était vu affecter l'extension BAK, reçoit à nouveau l'extension TXT. En d'autres termes, le fait de frapper ABORT annule tout ce que vous avez pu faire depuis le lancement de l'éditeur.

— La commande NEW permet

de travailler avec des fichiers tellement importants qu'ils ne peuvent rentrer en mémoire en une seule fois. Si vous êtes dans ce cas, vous lancerez l'éditeur sans vous soucier de quoi que ce soit ; celui-ci chargera tout ce qu'il pourra charger en mémoire, et vous pourrez ensuite travailler sur la portion ainsi chargée comme si de rien n'était. Lorsque vous allez faire appel à la partie qui n'a pu être chargée (lors d'une recherche de chaîne de caractères par exemple), l'éditeur vous indiquera qu'il ne trouve pas ce que vous lui demandez ; il vous suffit alors de vous placer sur la ligne de votre choix du fichier en cours d'édition et de frapper NEW. L'éditeur va sauvegarder tout ce qui se trouve avant la ligne sur laquelle vous vous êtes placé et va utiliser l'espace ainsi libéré pour charger la suite du fichier. Ce processus peut être répété autant de fois que nécessaire, sans risque quant à la cohésion du fichier. Attention ! Tout ce qui précède la ligne sur laquelle vous vous placez lors de la frappe du NEW étant sauvegardé sur disque, il n'est plus possible d'y faire appel. Si vous vous apercevez d'un oubli, il faut quitter l'éditeur puis y ré-entrer pour recommencer au début.

— commande FLUSH fonctionne de la même façon que la commande NEW mais, lorsqu'elle a sauvegardé ce qui se trouve avant la ligne sur laquelle vous vous êtes placé, elle ne lit plus rien sur le disque mais place l'éditeur en mode d'attente de commande ; cela permet, par exemple, d'ajouter au moyen d'un READ un autre fichier de grande taille au fichier en cours d'édition. Pour être franc, l'utilisation de cette commande est assez peu fréquente. Par contre, NEW vous sera vite utile si vous éditez des programmes de plusieurs K-octets en assembleur.

Adaptation à vos besoins

Un certain nombre de paramètres relatifs aux commandes SAVE, READ et WRITE peuvent être modifiés par vos soins pour adapter l'éditeur à

une situation particulière. Pour ce faire, il faut charger l'éditeur en mémoire sans le lancer avec un GET O.EDIT.CMD puis réaliser les modifications désirées ; un SAVE O.EDIT.CMD,0,0 permet ensuite de remettre celui-ci sur disque comme par le passé. Ces modifications sont relatives aux sous-programmes qui sont appelés lors d'un READ, d'un SAVE et d'un WRITE.

A l'adresse 16 et à l'adresse 17 se trouve l'adresse d'un sous-programme qui est appelé en début de chaque commande READ ; ce programme peut, par exemple, mettre en marche automatiquement un magnétophone en lecture ou toute autre fonction. Si vous ne voulez pas utiliser cette possibilité, il suffit de laisser en 16 et 17 le 0000 qui s'y trouve d'origine.

A l'adresse 18 et 19 se trouve l'adresse d'un sous-programme qui est appelé à la fin de chaque commande READ. Ce sous-programme peut, par exemple, arrêter un magnétophone automatiquement. Si vous ne voulez pas de cette possibilité, il suffit de laisser en 18 et 19 le 0000 qui s'y trouve d'origine.

A l'adresse 1A et 1B se trouve l'adresse d'un sous-programme qui est appelé au début de chaque commande SAVE ou WRITE et qui peut servir, par exemple, à mettre en marche un magnétophone en enregistrement. Dans la version d'origine de l'éditeur, ces adresses contiennent FD4C, qui est l'adresse d'un sous-programme de TAVBUG09 qui génère des caractères de synchronisation sur la cassette pour un meilleur fonctionnement de l'interface cassette. Si vous ne souhaitez pas utiliser cette possibilité, il suffit de mettre 0000 en 1A et 1B.

A l'adresse 1C et 1D se trouve l'adresse d'un sous-programme qui est appelé à la fin de toute commande WRITE ou SAVE. Si vous ne souhaitez pas utiliser cette possibilité, il vous suffit de laisser à ces adresses le 0000 qui s'y trouve d'origine.

A l'adresse 1E et 1F se trouve l'adresse du sous-programme de sortie de caractère

appelé lors d'un SAVE ou d'un WRITE. Dans la version d'origine, ces adresses contiennent FD37, qui est l'adresse du sous-programme de sortie d'un caractère sur l'interface cassette de TAVBUG09. Vous pouvez y mettre tout sous-programme correspondant à vos désirs, la seule contrainte étant que le caractère à sortir se trouve dans l'accumulateur A et qu'aucun registre du 6809 ne doit être détruit.

A l'adresse 20 et 21 se trouve l'adresse du sous-programme d'entrée de caractère appelé lors d'un READ. Dans la version d'origine, on y trouve FD46, qui est l'adresse du sous-programme d'entrée de caractère à partir de l'interface cassette de TAVBUG09. Vous pouvez mettre ce que vous voulez, mais il faut que le caractère rentré soit placé dans l'accum A et qu'aucun registre du 6809 autre que A ne soit modifié.

Ces possibilités de modifications sont intéressantes ; par exemple, si vous travaillez avec la carte IVG09 comme terminal, vous disposez sur la carte CPU09 d'une liaison série de libre. Cette liaison peut être raccordée à n'importe quel appareil en disposant (un autre micro-ordinateur, par exemple), et il est alors possible que l'éditeur lise des informations en provenance de cet appareil. Cette possibilité peut être mise à profit pour récupérer des programmes réalisés sur un autre système et dont les disquettes ne seraient pas compatibles. Les exemples peuvent être multipliés à l'infini, et nous faisons confiance à votre imagination pour trouver des applications à cette possibilité.

Le macro- assembleur

L'assembleur fourni d'origine avec le DOS est en réalité un macro-assembleur ; les connaisseurs apprécieront, les novices ne pourront apprécier qu'après avoir lu la notice ci-après.

La syntaxe d'appel de l'assembleur est, avec les conventions présentées le mois dernier : ASMB <FICHIER

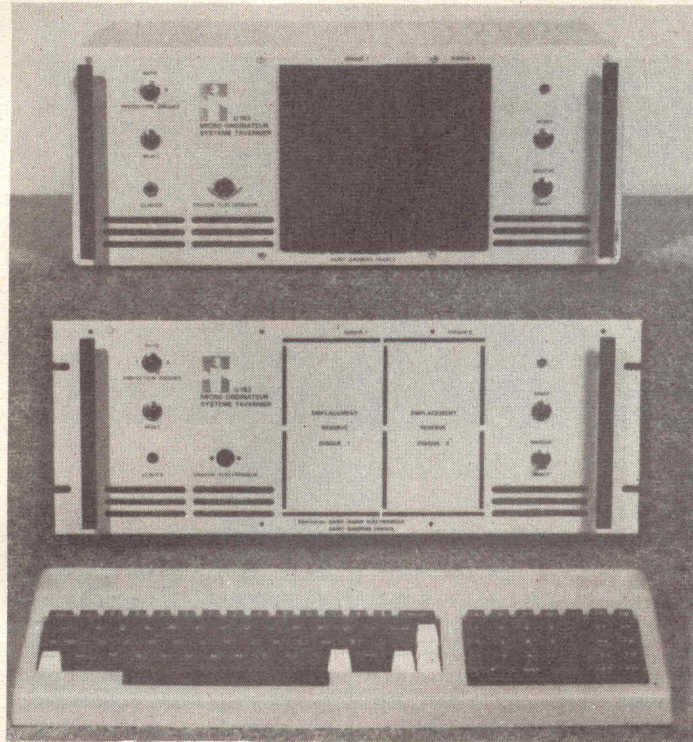


Photo 1. - Le boîtier proposé par Saint-Ignan Informatique avec et sans les lecteurs de disquettes.

SOURCE> (,<FICHIER OBJET> (,<+ OPTIONS>)), où FICHIER SOURCE est le nom du fichier à assembler dont l'extension par défaut est TXT et le lecteur par défaut celui de travail. Si FICHIER OBJET est spécifié, le programme assemblé sera mis sur disque sous ce nom de fichier. Si FICHIER OBJET n'est pas précisé, le nom du fichier source est repris mais affecté de l'extension BIN (pour binaire), et le programme assemblé est sauvegardé sous ce nom là. Des options peuvent être ajoutées sur cette ligne de commande en les faisant précéder d'un signe +. Nous allons en voir la liste ci-après mais, auparavant, voici quelques exemples de commandes d'assemblage valables : ASMB TOTO assemble le fichier TOTO.TXT pris sur le lecteur de travail et place le résultat en TOTO.BIN sur le lecteur de travail ; ASM O.TITI,1. PROG assemble le fichier TITI.TXT pris sur le lecteur 0 et place le résultat ainsi obtenu en PROG.BIN sur le lecteur 1.

Les options du macro- assembleur

Elles sont précisées sur la ligne de commande comme expliqué ci-avant et sont consti-

tuées d'une lettre par option. Si plusieurs options sont demandées simultanément, il faut placer toutes les lettres correspondantes les unes à la suite des autres, dans n'importe quel ordre mais sans signe ou espace entre les lettres.

— L'option B interdit la création du fichier binaire lors de l'assemblage ; cette option est utilisée en général avant l'assemblage définitif d'un programme pour voir sur le listing les messages d'erreurs éventuels.

— L'option L interdit la sortie d'un listing d'assemblage ; seul le fichier binaire est créé (si l'option B n'a pas été spécifiée). Les lignes conduisant à un message d'erreur de l'assembleur sont tout de même visualisées malgré cette commande.

— L'option S supprime l'impression de la table des étiquettes en fin de listing. Si cette option n'est pas spécifiée, tous les symboles utilisés sont imprimés en fin de listing par ordre alphabétique avec la valeur qui leur a été affectée par l'assembleur.

— L'option G interdit l'impression de lignes multiples au niveau des directives FCC, FCB, FDB, et ne laisse apparaître sur le listing que la ligne contenant la directive (voir plus avant pour la signification de ces sigles).

— L'option N fait imprimer sur le listing les numéros de lignes ; ceux-ci étant identiques à ceux utilisés par l'éditeur, cela peut faciliter la correction d'éventuelles erreurs. Même si cette option n'est pas demandée, les messages d'erreur sont toujours affichés avec un numéro de ligne pour la raison exposée ci-avant.

— L'option Y efface automatiquement le fichier binaire de même nom que celui que va produire l'assembleur et qui pourrait déjà exister sur le disque. En effet, si cette option n'est pas spécifiée et que vous demandiez un assemblage conduisant à un nom de fichier déjà présent sur la disquette (nom de fichier qui pourrait résulter, par exemple, d'un assemblage précédent), l'assembleur vous demande l'autorisation d'effacer le fichier déjà existant ; si vous répondez non, l'assemblage demandé n'a pas lieu. Attention, vous ne pouvez répondre à cette question que par O pour oui ou N pour non ; nous avons en effet oublié d'y inclure le Y pour yes !

— L'option D interdit l'impression de la date qui a lieu en haut de chaque page de listing si vous avez spécifié l'option PAG dans la source de votre programme (voir plus avant la signification de PAG).

— L'option W interdit l'impression des « warnings », ou avertissements en français. En effet, l'assembleur détecte les erreurs d'assemblage et vous les signale, mais il peut aussi détecter des points de listings qui ne sont pas forcément des erreurs mais où votre attention doit être appelée ; par exemple si vous utilisez un LBSR à un endroit où vous pourriez mettre un BSR, votre programme fonctionnera quand même, mais vous aurez utilisé une instruction trop performante (et plus longue que celle que vous auriez pu mettre) ; l'assembleur vous l'indiquera sauf si vous demandez l'option W. Attention, alors qu'un listing comportant des messages d'erreur ne conduit généralement pas à un fichier binaire utilisable, car certaines lignes de programme n'ont pu être assemblées correctement, un listing contenant des « war-

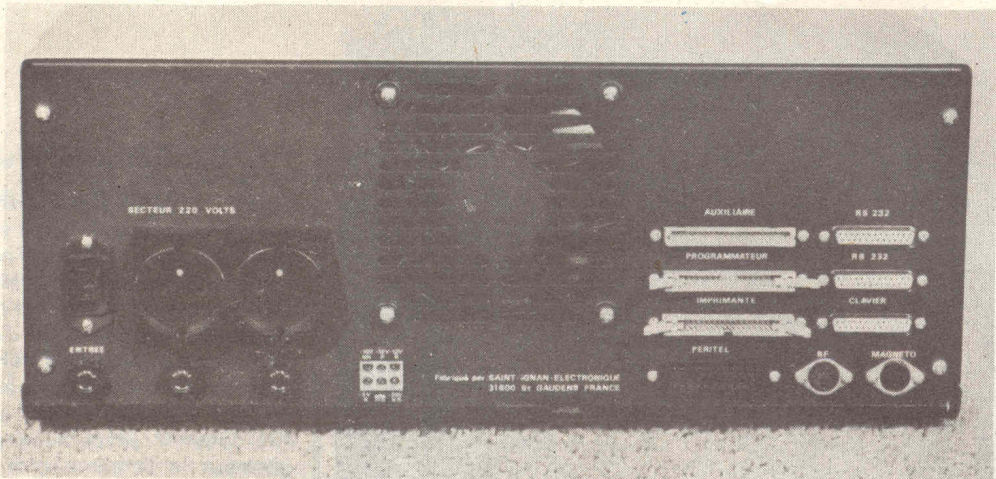


Photo 2. — La face arrière de ces boîtiers.

nings » est généralement utilisable sans problème.

— L'option P permet de spécifier un numéro de page à partir duquel commencera l'impression du listing. Son utilisation se fait de la façon suivante : PX, où X est un nombre décimal représentant le numéro de page où commencer. X doit être compris entre 1 et 65535 et doit être suivi par une virgule ou un espace exclusivement ; cela implique que cette option, si elle est demandée, soit la dernière de la liste des options.

Sortie de listing sur imprimante

Comme toutes les autres commandes du DOS vues le mois dernier, cet assembleur ne dispose pas en interne d'un programme de sortie sur imprimante ; il faut faire appel à la commande P du DOS sous la forme P ASMB, etc., selon la procédure expliquée le mois dernier pour celle-ci. Tout ce que produira alors l'assembleur, compte tenu des options que vous avez demandées, sera dirigé sur l'imprimante plutôt que sur le terminal du système.

La syntaxe assembleur

Si vous suivez nos articles d'initiation à la micro-informatique conjointement à ceux-ci, vous devez commencer à savoir que l'on ne peut fournir à l'assembleur un programme frappé n'importe comment ; il faut respecter une certaine syntaxe pour que l'assembleur distingue les étiquettes des

instructions ou des commentaires. C'est cette syntaxe que nous allons décrire maintenant. Précisons qu'elle est commune à de nombreux assembleurs et qu'elle est, en particulier, compatible des assembleurs proposés par Motorola et Thomson Efcis sur leurs systèmes de développement (Exorciser et Thémis).

Cet assembleur est un assembleur à deux passes, c'est-à-dire qu'il lit deux fois la source du programme ; la première fois, il construit une table de toutes les références symboliques (étiquettes) et, la deuxième fois, il assemble réellement.

Une ligne source, c'est-à-dire une ligne de ce que vous allez assembler (revoir si nécessaire notre article d'initiation du mois de janvier), est composée de quatre champs :

— Le champ étiquette ou symbole qui doit commencer au premier caractère de la ligne par une lettre majuscule ou minuscule. Si la ligne ne comporte pas ce champ, elle doit commencer par un espace. Les étiquettes peuvent contenir des lettres majuscules et minuscules et des chiffres de 0 à 9, ainsi que le symbole « souligné » (à ne pas confondre avec le tiret utilisé, par exemple, pour les mots composés). Les majuscules et les minuscules ne sont pas équivalentes ; ainsi, l'étiquette XYZ est différente de xyz. Une étiquette doit impérativement commencer par une lettre. La longueur d'une étiquette peut être quelconque, mais l'assembleur ne prend en compte que les six premiers caractères. Attention donc : INITIALISATION et INITIAL seront pour vous des étiquettes différentes,

alors que l'assembleur les tronquera à INITIA (six caractères) et les verra donc comme la même chose ; il vaut donc mieux prendre l'habitude de n'utiliser que des étiquettes de six caractères au maximum. Une étiquette doit être terminée par un espace (si elle se trouve en début de ligne et qu'elle est suivie par un des champs décrits ci-après) ou par un retour chariot si elle constitue le dernier champ d'une ligne. Enfin, une étiquette ne doit pas être le nom d'un registre du 6809 ce qui vous interdit : A, B, CC, DP, X, Y, U, S, D, PC comme étiquettes, ce qui est logique, car cela peut conduire très vite à des confusions importantes.

— Le champ instruction ou pseudo instruction contient une des instructions du 6809 ou une des pseudo instructions reconnues par l'assembleur et décrites ci-après. Ce champ est composé de lettres majuscules ou minuscules mais celles-ci sont équivalentes ; ainsi LDA et lda sont comprises toutes deux comme LDA par l'assembleur. Des chiffres de 0 à 9 peuvent aussi être présents dans ce champ. Ce champ instruction est terminé par un espace s'il est suivi par une opérande, ou par un espace ou un retour chariot s'il n'est suivi par aucune opérande.

— Le champ opérande suit toujours une instruction ou pseudo instruction dont il est séparé par un espace, comme dit ci-avant à propos du champ instruction. Il peut contenir de nombreuses informations différentes puisque cela dépend de l'instruction ou de la pseudo instruction qui le précède ; en particulier, et comme expliqué

en détail ci-après, on peut y trouver : des étiquettes (qui répondent alors aux normes exposées ci-avant pour le champ étiquettes), des registres, des constantes numériques, des caractères alphanumériques, des expressions mathématiques, etc. Les seules contraintes relatives à ce champ sont qu'il ne doit pas contenir d'espace, qu'il doit être terminé par un espace ou un retour chariot.

— Le champ commentaire permet de mettre des commentaires sur les lignes de listing ; il est optionnel mais, s'il est présent, il doit respecter les règles suivantes : il doit commencer par un espace (celui qui doit terminer le champ opérande ou le champ instruction, par exemple), il peut contenir tous les caractères ASCII de code compris entre 20 et 7F, c'est-à-dire tous les caractères ASCII imprimables, il doit se terminer par un retour chariot, car il est impérativement le dernier champ d'une ligne.

Pour conclure cette présentation des divers champs d'une ligne respectant la syntaxe assembleur, nous vous présentons en figure 1 quelques lignes correctes avec la représentation des divers champs et séparateurs.

Un seul type de ligne admise par l'assembleur fait exception aux règles ci-avant : c'est la ligne de commentaires. Il est en effet possible de placer en n'importe quel endroit d'un programme des lignes dites de commentaires ; leur contenu peut être quelconque (commentaires, mais aussi motifs décoratifs de présentation du listing, nom de société, etc.), la seule condition étant que ces lignes commencent impérativement par une astérisque placée en premier caractère de la ligne concernée.

Registres et expressions

De très nombreuses instructions font appel aux noms des registres du 6809 ou à des expressions au sens large. Nous allons voir ci-après comment sont définis ces éléments. Les registres (voir si nécessaire nos articles d'initia-

tion) sont représentés par les appellations suivantes :

- A, B, D pour les accumulateurs A, B, D.
- X et Y pour les deux index X et Y.
- U et S pour les deux pointeurs de piles utilisateur et système.
- CC pour le registre d'état ou de codes, conditions que l'on trouve aussi sous l'appellation CCR dans certaines fiches techniques.
- DP pour le registre de page directe, que l'on trouve aussi sous l'appellation DPR dans certaines fiches techniques.
- PC pour le compteur ordinal (le Program Counter).

Les expressions sont constituées par des données combinées entre elles au moyen d'opérateurs arithmétiques, logiques, de relation ou de décalage.

Ces données peuvent être constituées par :

- Des constantes numériques qui seront converties par l'assembleur en entiers codés sur 16 bits. S'ils sont plus grands que ce que peut admettre un tel codage, ils seront tronqués en conséquence, et cela sera signalé par un avertissement sur le listing (si l'option W n'a pas été mise en place). Ces constantes numériques peuvent être exprimées en décimal, binaire, hexadécimal et octal. Il faut les faire précéder d'un préfixe en conséquence, comme indiqué dans le tableau de la figure 2 ; ainsi la valeur 10 considérée en décimal sera écrite 10, en hexadécimal elle serait notée \$10 et en octal 10.

- Des caractères alphanumériques qui, sous réserve qu'il soient précédés par une apostrophe, seront convertis par l'assembleur en leur code ASCII. Ainsi 'A sera converti en \$41, puisque le code ASCII de A est 41. Tous les caractères imprimables sont autorisés.
- Des étiquettes qui respectent les contraintes indiquées ci-avant lors de la description du champ étiquette. Ces étiquettes sont alors remplacées par l'assembleur par la valeur qui leur a été affectée lors de leur définition.
- Un caractère particulier qui est l'astérisque et qui représente le compteur ordinal (PC).

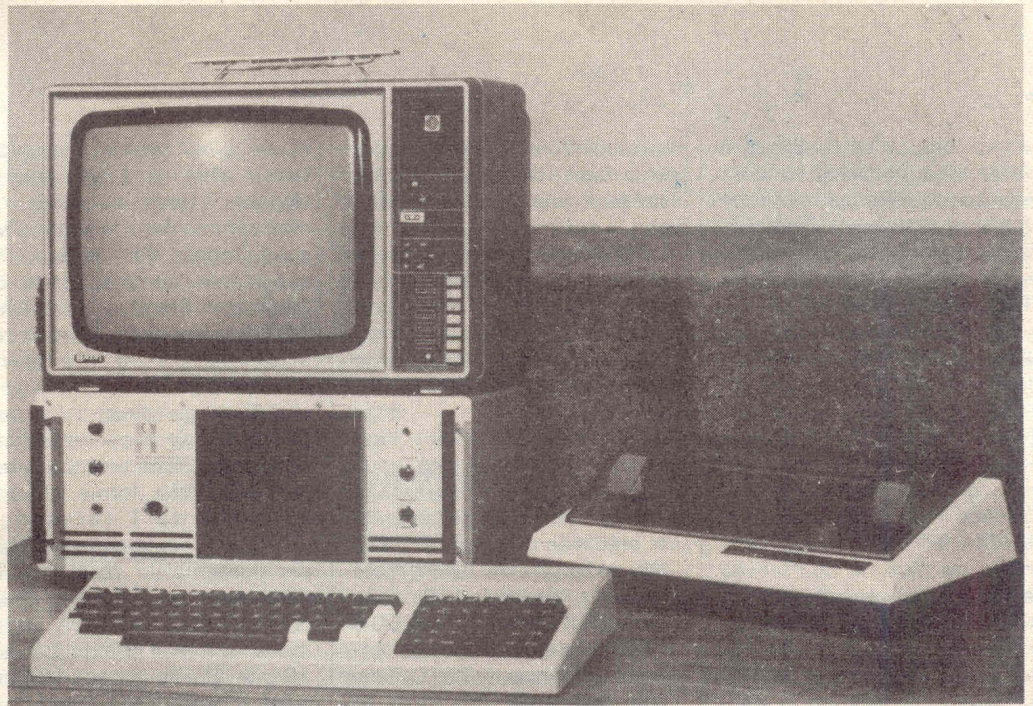


Photo 3. - Vue du système dans une configuration « confortable ».

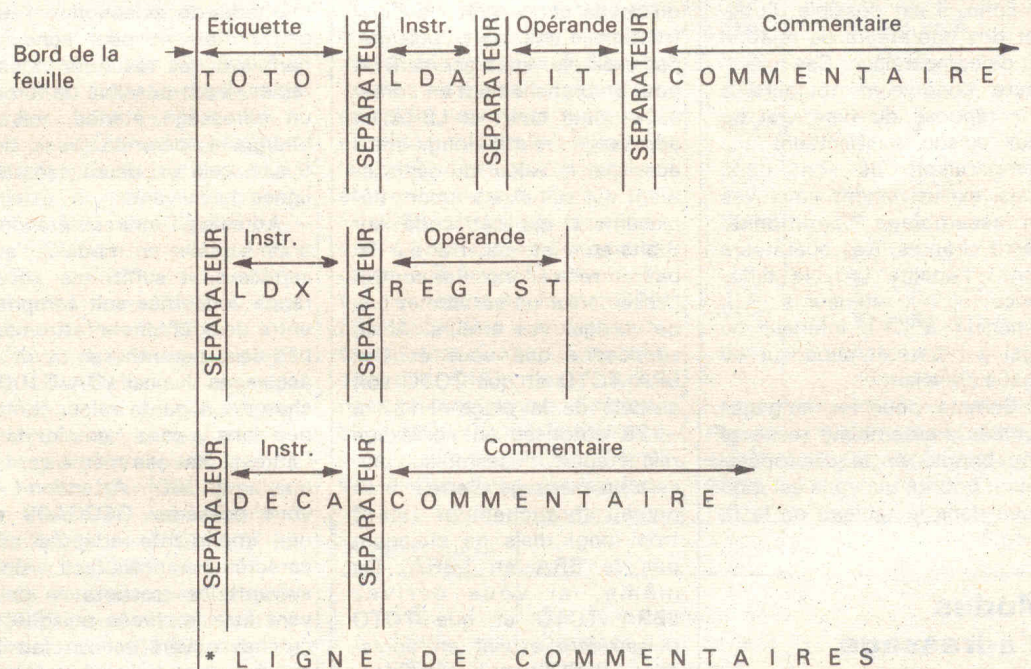


Fig. 1. - Quelques lignes respectant la syntaxe assembleur

Ces données peuvent être combinées au moyen d'opérations arithmétiques qui sont : l'addition, la soustraction, la multiplication et la division entière (c'est-à-dire que si les nombres ne se divisent pas en entiers, le reste est ignoré). Le calcul de l'expression est réalisé au moment de l'assemblage par l'assembleur et ne doit pas être confondu avec le programme à assembler proprement dit. Un exemple d'utilisation de cette possibilité est le suivant : soit un circuit d'interface type PIA, par exemple,

Base	Symbole	Exemple
Décimal	sans	256
Binaire	%	%11001010
Octal	@	@ 26
Hexadécimal	\$	\$ FE

Fig. 2. - Symboles relatifs aux diverses bases

dont l'adresse du premier registre interne est définie par l'étiquette PIA ; les autres registres du PIA (qui se trouvent à des adresses immédiatement supérieures d'une unité à chaque fois) pourront être définis

par PIA+1, PIA+2, PIA+3, etc. (même si l'imprimerie en ajoute, il n'y a aucun espace dans ces expressions !).

Des opérateurs logiques peuvent aussi être utilisés dans les expressions ; ils travaillent

sur 16 bits, c'est-à-dire que l'assembleur convertit les données concernées en mots de 16 bits et réalise ensuite l'opération logique sur ces mots. Les opérations logiques sont le ET logique représenté par un & ; le OU logique représenté par un @ ; le complément logique représenté par un ! ; le décalage à droite représenté par un >> et le décalage à gauche représenté par un <<. Pour ces deux derniers opérateurs, le décalage fait perdre les bits décalés et fait entrer des zéros ; de plus, le nombre de décalages est spécifié de la façon suivante : TOTO>>N, où N représente le nombre de décalages (à droite dans ce cas) qui seront réalisés sur la donnée dont la valeur est affectée à l'étiquette TOTO.

Enfin, il est possible d'utiliser des opérateurs de relation ou de comparaison. Ces opérateurs conduisent toujours à une réponse du type vrai ou faux puisqu'ils effectuent une comparaison. Ils sont donc quasi exclusivement employés en assemblage conditionnel décrit ci-après. Ces opérateurs sont : l'égalité (=), la différence (<>), inférieur à (<), supérieur à (>), inférieur ou égal à (<=) et supérieur ou égal à (>=).

Comme pour les langages évolués, l'assembleur respecte une priorité entre ces opérateurs, priorité qui vous est indiquée dans le tableau de la figure 3.

Modes d'adressage

Nous n'allons pas ici passer en revue les modes d'adressage du 6809, qui ont été vus en détail dans un article de notre série d'initiation à la micro-informatique, auquel nous vous demandons de vous reporter en cas de problème.

Nous allons seulement insister sur la façon de signaler à l'assembleur quel mode vous utilisez.

— Adressage inhérent : vous n'avez aucune indication à donner puisque ce mode d'adressage n'en est pas vraiment un mais fait plutôt partie de certaines instructions particulières.

— Adressage immédiat : l'assembleur considère que vous faites appel à de l'adressage immédiat lorsque l'opérande est précédée du symbole dièse (#) ; ainsi LDA #10 chargera la valeur décimale 10 dans l'accumulateur A et LDB #TOTO chargera la valeur affectée à l'étiquette TOTO dans l'accumulateur B.

— Adressage relatif court et long : l'assembleur est informé du mode choisi grâce à l'instruction d'une part, puisque, par exemple, vous écrirez BRA pour un branchement en adressage relatif court et LBRA en adressage relatif long, mais aussi par la valeur du déplacement qui suit l'instruction, déplacement qui est codé sur 8 bits en relatif court et sur 16 bits en relatif long. Par contre, l'assembleur ne se permet pas de corriger vos erreurs. Ainsi, supposons que vous écriviez BRA TOTO et que TOTO soit au-delà de la plage +127 à -128 autorisée en adressage relatif court, l'assembleur placera un message d'erreur à ce niveau (branchement relatif trop long) mais ne changera pas le BRA en LBRA. De même, si vous écriviez LBRA TOTO et que TOTO puisse être atteint en adressage relatif court, l'assembleur ne changera pas le LBRA en BRA mais placera un avertissement à ce niveau (un avertissement et non une erreur, car cela n'empêchera pas le programme de fonctionner).

— Adressage étendu : comme c'est le mode le plus courant

et le plus simple d'emploi, vous n'avez rien à faire pour le signaler ; toute opérande ne rentrant pas dans le cadre d'un autre mode d'adressage est considérée comme étant en adressage étendu ; ainsi LDA TOTO chargera A par le contenu de la mémoire d'adresse TOTO.

— Adressage direct : lui non plus n'a pas besoin d'indication particulière puisque c'est en réalité une forme un peu spéciale de l'adressage étendu. La seule chose utilisée par l'assembleur pour faire la différence entre ces deux modes est le fait que l'opérande soit sur 8 bits ou sur 16 bits. Si elle est sur 16 bits, c'est un adressage étendu ; si elle est sur 8 bits, c'est un adressage direct. Comme cette méthode de sélection est un peu brutale et peut conduire parfois à des résultats indésirables, il est possible de forcer un adressage étendu, même lorsque l'opérande tient sur 8 bits, cela est décrit dans les lignes qui suivent.

— Adressage indirect étendu : pour signaler ce mode à l'assembleur, il suffit que l'opérande concernée soit comprise entre deux crochets (attention, pas deux parenthèses ou deux accolades !) ainsi LDA \$1000 chargera A par la valeur contenue dans la case mémoire dont l'adresse est elle-même contenue en [1000]. Attention ! si vous possédez GCGGA09 et une imprimante adaptée aux caractères français (voir article suivant) les crochets ne peuvent être imprimés puisque le crochet ouvert est en fait le symbole ° et le crochet fermé le symbole § ; cela n'a aucune importance au niveau assembleur qui, lui, interprète correctement les codes ; il n'y a que sur votre listing que vous lirez ° et § au lieu des crochets.

— Adressage indexé : il en existe de multiples formes, comme nous l'avons vu lors de son étude dans nos articles d'initiation ; nous allons les passer en revue, mais vous allez vite constater que les notations adoptées sont parfaitement logiques.

— Adressage étendu avec déplacement constant : il suffit d'écrire l'opérande sous la forme D,R, où D représente le

déplacement (qui peut être une valeur numérique mais aussi une étiquette) et R le registre à utiliser comme index, puisque X, Y, U et S peuvent employer ce mode d'adressage. Ainsi, LDA 5,X, LDB TOTO,U, LDX O,Y, etc., sont corrects.

— Adressage indexé avec accumulateur comme déplacement : il suffit d'employer la notation précédente, où D devient un des accumulateurs A, B ou D.

— Adressage indexé auto-incrémenté : la notation en est la suivante : O,R+ ou ,R+ ou O,R++ ou ,R++. En effet, ce mode d'adressage fait toujours appel à un déplacement nul, d'où le O et le fait que, pour vous économiser de la frappe, l'assembleur vous autorise même à omettre le O. De plus, ce mode existe avec une incrémentation de 1 ou de 2 matérialisée par un signe + ou deux signes +. Le fait que les signes + soient placés à la fin de l'opérande est un moyen mnémotechnique de vous rappeler que ce mode est en fait un mode POST incrémenté, c'est-à-dire que l'incrémentation du registre se fait après l'exécution de l'instruction concernée par ce mode d'adressage.

— Adressage auto-décrémenté : la notation est tout aussi logique que pour le mode précédent et il faut écrire : O,-R ou ,-R ou O,- -R ou - -R, où R représente le registre concerné. Un signe - signifie une décrémentation d'une unité, et deux signes - de deux unités. Le fait de placer les signes avant le registre aide à se souvenir que c'est un mode PRE décrémentation, c'est-à-dire que la décrémentation a lieu avant l'exécution de l'instruction concernée par ce mode.

— Adressage indexé indirect : tous les modes précédents peuvent être utilisés en mode indirect ; il suffit, comme pour le mode étendu indirect, de les entourer par deux crochets ; ainsi LDA [5,X] sera un mode indexé avec déplacement constant indirect. Attention, si les modes auto-incrémenté ou auto-décrémenté sont utilisés avec une indirection, l'incrémentation ou la décrémentation doit impérativement être de deux et non de un puisque

1	Expressions entre parenthèses
2	Signes + et -
3	Opérateurs de décalage
4	Multiplication et division
5	Addition et soustraction
6	Opérateurs de comparaison
7	Complément logique
8	ET et OU logiques

Fig. 3. — Priorité des opérateurs, le 1 est le plus prioritaire.

l'indirection fait que l'on va chercher une adresse et non une donnée et que les adresses sont codées sur 16 bits et nécessitent donc deux cases mémoire.

— Adressage relatif par rapport au PC : ce mode d'adressage est tout simplement signalé à l'assembleur par le fait que l'opérande est suivi par une virgule et PCR ; ainsi LDA TOTO,PCR fera prendre TOTO en adressage relatif par rapport au PC.

Cette revue rapide des divers modes d'adressage n'est pas, rappelons-le, un cours sur les modes d'adressage du 6809, puisque cela a fait l'objet d'un de nos articles d'initiation, mais seulement une présentation de la syntaxe utilisée par l'assembleur pour reconnaître ceux-ci.

A propos des messages d'avertissement concernant les modes d'adressage relatifs et pour alléger un peu la présentation des listings, toute utilisation d'un adressage relatif « trop » long (LBXX au lieu de BXX, ou JMP au lieu d'un BRA) n'est pas indiquée par un avertissement écrit « en clair » mais simplement par une petite flèche placée en premier caractère de la ligne concernée.

Le forçage d'un mode d'adressage

Ainsi que nous l'avons expliqué ci-avant, le seul point pris en compte par l'assembleur pour savoir s'il doit travailler en adressage direct ou étendu est la taille de l'opérande. Si elle est sur 8 bits, c'est du direct, si elle est sur 16, c'est de l'étendu. Comme cela conduit parfois à des erreurs, vous pouvez forcer un mode ou l'autre. Pour forcer un mode étendu, il suffit de faire précéder l'opérande du symbole > ; ainsi LDA >TOTO fera utiliser l'adressage étendu quelle que soit la taille de TOTO. Pour forcer un mode direct il suffit de faire précéder l'opérande du symbole < ainsi LDA <TOTO fera utiliser l'adressage direct quelle que soit la taille de TOTO. Dans ce dernier cas, l'assembleur considère que les 8 bits de poids

fort de l'adresse TOTO sont contenus dans le DPR, et il code uniquement les 8 bits de poids faible.

Pour éviter toute confusion, il faut impérativement que l'opérande précédée de ce symbole soit la première d'une expression sinon l'assembleur considérera ce symbole comme un opérateur de relation, et cela aura des résultats totalement différents de ceux attendus.

Les instructions

Nous n'allons pas donner ici la liste des instructions 6809 qui a été publiée sous forme de tableau résumé et qui a été décrite ensuite dans nos articles d'initiation. Nous allons nous limiter à quelques remarques quant à la syntaxe.

Toutes les instructions faisant appel à un registre doivent être écrites accolées au registre (LDA, CMPA, etc.), mais un espace est aussi toléré entre l'instruction et le nom du registre si cette instruction existait déjà en 6800 ; ainsi l'on doit écrire, en 6809, CMPA, mais l'assembleur comprend aussi CMP A, qui était la syntaxe 6800.

Toutes les instructions qui font appel à plusieurs registres sont séparées de ceux-ci par un espace et les registres intervenant sont séparés les uns des autres par des virgules. Par exemple, vous écririez : TFR A,DP ou EXG A,B ou PSHS A,B, CC,DP, etc.

Les mnémoniques propres au 6800 et qui existent en tant qu'instructions 6809, mais qui ne s'écrivent plus de la même façon, sont aussi comprises par cet assembleur. Ainsi comprend-il : LDAA, LDAB, STAA, STAB, CPX, qu'il interprète respectivement par LDA, LDB, STA, STB et CMPX.

Les instructions typiquement 6800 qui n'existent plus en 6809 sont aussi comprises par cet assembleur qui les traduit automatiquement par la — ou les — instructions 6809 donnant la même fonction. La figure 4 donne la liste de ces instructions typiquement 6800.

Enfin, et comme si ce n'était pas suffisant, cet assembleur comprend aussi des

mnémoniques du 6801 et les remplace par la — ou les — instructions 6809 qui accomplissent la même fonction ; ces mnémoniques sont indiqués figure 5. De plus, il autorise aussi des mnémoniques de « confort » qui sont plus « parlants » que ceux d'origine et qui sont, eux, présentés figure 6.

Les directives de l'assembleur

Ces directives, que l'on appelle aussi pseudo-instructions, ont plusieurs fonctions, mais, quelles que soient celles-ci, ces directives ne doivent pas être confondues avec des instructions du programme à assembler. Elles n'ont une action qu'au niveau de l'assembleur pour lui donner certaines indications relatives à des étiquettes ou à des actions à accomplir. Leur liste complète est résumée dans le tableau de la figure 7, et nous allons en étudier le rôle ci-après.

— ORG : est utilisée pour indiquer à l'assembleur l'origine de ce qui suit cette directive ; elle s'utilise sous la forme ORG EXPRESSION à l'exclusion de toute autre forme. Si aucune directive ORG n'est placée dans un programme, l'assembleur utilise 0000 par défaut. Il peut y avoir autant de directives ORG que vous voulez dans

un programme et leur ordre peut être quelconque (un ORG \$100 peut apparaître après un ORG \$2000 sans pour cela que l'assembleur se trompe).

— END : est utilisée pour indiquer à l'assembleur qu'il a atteint la dernière ligne du programme à assembler. Cette

	Mnémonique Fonction
ABA	A + B → A
CBA	Compare B et A
CLC	0 → C
CLI	0 → I
CLV	0 → V
DES	S - 1 → S
DEX	X - 1 → X
INS	S + 1 → S
INX	X + 1 → X
SBA	A - B → A
SEC	1 → C
SEI	1 → I
SEV	1 → V
TAB	A → B
TAP	A → CCR
TBA	B → A
TPA	CCR → A
TSX	S → X
TXS	X → S
WAI	Attente d'interruption

Fig. 4. — Instructions 6800 comprises par l'assembleur 6809

Mnémoniques	Fonction
ASLD	Décalage à gauche de D
LSRD	Décalage à droite de D
PSHX	Pousse X sur la pile
PULX	Tire X de sur la pile
LDAD	M → D
STAD	D → M

Fig. 5. — Instructions 6801 comprises par l'assembleur 6809

Mnémoniques	Fonction
BEC ou LBEC	Branchement si pas d'erreur
BES ou LBES	Branchement si erreur
CLF	0 → F (masque de FIRO)
CLZ	0 → Z
SEF	1 → F
SEZ	1 → Z

Fig. 6. — Instructions de « confort » comprises par l'assembleur 6809

directive est facultative car l'assembleur peut s'arrêter seul lorsqu'il ne trouve plus rien dans le fichier à assembler. Une autre utilisation de cette directive est de permettre de donner au fichier ainsi assemblé une adresse de transfert (voir la commande SAVE de la notice du DOS), et ce automatiquement ; il suffit de terminer le programme par END EXPRESSION, où expression est une étiquette ou la valeur de l'adresse de début du programme, pour que le fichier BIN créé par l'assembleur soit muni de cette valeur comme adresse de transfert.

— RMB : permet de réserver des octets en mémoire et de leur donner un nom sous forme d'une étiquette ; l'utilisation en est la suivante : (ETIQUETTE) RMB <EXPRESSION>, où ETIQUETTE est l'étiquette qui correspondra à — ou au premier — des octets ainsi réservé(s) et où EXPRESSION indique le nombre d'octets à réserver. Ainsi TOTO RMB 2 réserve 2 octets de mémoire, le premier s'appelant TOTO. Plusieurs utilisations sont possibles lors de réservations de multiples octets selon votre façon de travailler ; ainsi, dans l'exemple ci-avant, les 2 octets réservés pourront être appelés par TOTO pour le premier et par TOTO+1 pour le second, mais vous auriez aussi pu faire TOTO RMB 1 puis, à la ligne suivante, TITI RMB 1. Le premier octet se serait encore appelé TOTO mais le second

se serait appelé TITI. Attention, RMB ne fait rien sur les octets réservés, c'est-à-dire qu'ils ne doivent pas être considérés comme contenant une valeur prédéfinie.

— FCB : permet de placer en mémoire des constantes codées sur 8 bits. L'utilisation en est la suivante : (ETIQUETTE) FCB <EXPRESSION1>(<EXPRESSION2>..., <EXPRESSIONN>), où EXPRESSION1 à EXPRESSIONN sont des expressions qui seront tronquées (si nécessaire) pour tenir sur 8 bits. Ainsi, TOTO FCB \$10,25 placera 10 (hexadécimal) en mémoire, et cette mémoire aura pour adresse TOTO et placera 25 (décimal) dans la case mémoire d'adresse suivante (qui pourra alors être référencée par TOTO+1 par exemple).

— FDB : a le même rôle que FCB mais place en mémoire des mots de 16 bits. La syntaxe est la même que pour FCB mais les expressions sont tronquées (si nécessaire) à 16 bits. Le placement en mémoire a lieu conformément aux règles propres aux microprocesseurs Motorola, à savoir poids forts à l'adresse n, poids faibles à l'adresse n+1. Si l'expression qui suit le FDB tient sur 8 bits, l'octet de poids fort est mis à 00 mais en aucun cas l'assembleur ne transforme le FDB en FCB. Aucun message n'est généré dans ce cas puisque vous pouvez très bien décider de coder sur 16 bits des mots de 8 bits (si vous placez, par

exemple, en mémoire une suite d'adresses, ce qui impose un codage sur 16 bits, et que celles-ci soit inférieures à \$FF, ce qui permet de les coder sur 8 bits).

— FCC : a le même rôle que FCB et utilise la même syntaxe mais en n'étant pas suivie par des expressions mais par une ou plusieurs chaînes de caractères comprises entre deux délimiteurs identiques mais quelconques. Les caractères compris entre ces délimiteurs sont alors convertis en leurs codes ASCII et sont placés les uns à la suite des autres en mémoire. Les délimiteurs peuvent être n'importe quels caractères non alphanumériques. Une même chaîne doit être comprise entre deux délimiteurs identiques mais deux chaînes présentes sur une même ligne peuvent utiliser des délimiteurs différents. De plus, des valeurs hexadécimales (donc impérativement précédées du symbole dollar) peuvent être placées sur des lignes de chaînes de caractères après un FCC, la valeur hexadécimale sur 8 bits est placée sans altération. Dans ce cas, ou dans le cas de plusieurs chaînes, sur une même ligne, des virgules doivent être utilisées comme séparateurs entre les chaînes. Par exemple, TOTO FCC ?ABC? placera 41, 42, 43 (hexadécimal) qui sont les ASCII de A, B, C en mémoire à partir d'une adresse qui sera repérée par TOTO. TOTO FCC \$10,/ ABCD/,\$2,*A* placera en mémoire 10, 41, 42, 43, 02, 41, à partir d'une adresse qui sera repérée par TOTO.

— EQU : permet de donner une valeur à une étiquette de façon permanente c'est-à-dire qu'une fois cette directive écrite pour une étiquette donnée, elle ne doit pas apparaître à nouveau ailleurs pour la même étiquette. L'utilisation est très simple puisqu'il suffit d'écrire ETIQUETTE EQU EXPRESSION pour affecter à étiquette la valeur de l'expression ; ainsi TOTO EQU \$1000 donnera la valeur hexadécimale (à cause du dollar) 1000 à l'étiquette TOTO.

— SET : joue un rôle analogue à EQU puisque cette directive permet aussi de donner une

valeur à une étiquette, mais cette affectation de valeur est temporaire et plusieurs directives SET peuvent apparaître dans le même listing. La valeur affectée à l'étiquette ainsi définie à un instant donné est celle de la dernière directive SET rencontrée. L'utilisation est identique à EQU, à savoir : ETIQUETTE SET EXPRESSION.

— REG : permet d'affecter une étiquette à une liste de registres qui revient souvent dans un programme. En effet, en 6809, certaines instructions telles que PSHS, PULS, PSHU, PULU, peuvent travailler sur plusieurs registres qui sont alors spécifiés après l'instruction ; ainsi, pour sauvegarder des registres sur la pile S, écrivons-nous PSHS A,B,X,Y, par exemple. Si une liste donnée revient souvent et pour vous économiser du travail, il est possible de lui donner une étiquette au moyen de cette directive de la façon suivante : ETIQUETTE REG LISTE DE REGISTRES, où LISTE DE REGISTRES est constituée par les noms des registres concernés, tels qu'ils ont été définis ci-avant, séparés par des virgules. Si nous reprenons notre exemple, nous pourrions écrire : TOTO REG A,B,X,Y qui s'utilisera alors sous la forme PSHS #TOTO. Attention, le symbole dièse est obligatoire dans ce cas toutes les fois qu'une liste de registres est définie au moyen d'une étiquette.

— SETDP : permet d'indiquer à l'assembleur la valeur contenue dans le registre de page direct (DP ou DPR) pour que celui-ci puisse savoir pour quelles adresses il doit employer l'adressage direct. Ainsi, si vous avez placé \$10 dans le DP, lorsque vous aurez fait un SETDP \$10, l'assembleur saura que toutes les adresses comprises entre 1000 et 10FF peuvent être référencées en adressage direct. L'utilisation est fort simple : SETDP VALEUR, où VALEUR est le mot de 8 bits contenu dans le DP. Autant de SETDP que nécessaire peuvent apparaître dans un même programme ; à un instant donné, l'assembleur considère comme valide le dernier rencontré. Si aucune directive SETDP n'est

Directive	Fonction
ORG	Définition d'une origine
END	Fin d'un programme
RMB	Réserve mémoire
FCB	Définition d'une constante 8 bits
FDB	Définition d'une constante 16 bits
FCC	Définition d'une constante ASCII
EQU	Affectation d'une valeur (définitive)
SET	Affectation d'une valeur (provisoire)
REG	Définition d'une liste de registre
SET DP	Affectation d'une valeur au DPR
PAG	Saut page du listing
SPC	Saut ligne du listing
NAM ou TTL	Définition d'un titre
STTL	Définition d'un sous-titre
ERR	Définition d'un message d'erreur
RPT	Répétition d'une ligne
LIB	Accès à un fichier disque
OPT	Définition d'options

Fig. 7. — Liste et fonctions des directives de l'assembleur

utilisée dans un programme, l'assembleur considère que le DP contient 00 et travaille donc en adressage direct de 00 à FF (ceci afin d'être compatible avec le 6800).

— PAG : permet de faire sauter une page au niveau du listing sous réserve que l'option pagination ait été demandée (voir ci-après). Cette directive n'est pas imprimée sur le listing sauf en cas d'erreur au niveau de son utilisation. Il est possible, après le saut page ainsi demandé, de faire continuer la numérotation des pages du listing à la valeur de votre choix en faisant suivre PAG d'une expression dont la valeur sera ce nouveau numéro.

— SPC : permet de faire sauter des lignes sur le listing et s'utilise de la façon suivante : SPC N(M), où N représente le nombre de lignes à faire sauter et où la valeur optionnelle M permet de spécifier le nombre de lignes que vous souhaitez garder ensemble sur une page. Si le nombre de sauts de lignes demandé ne permet pas de respecter cette valeur, un saut page est automatiquement généré. Cette utilisation de M est cependant assez rare. Cette directive, tout comme PAG, n'est pas imprimée sur le listing. Cette directive est optionnelle si vous ne souhaitez sauter que quelques lignes ; en effet, les lignes vides de tout caractère fournies à l'assembleur sont traduites au niveau du listing par des lignes vides également. Donc, lorsque vous frappez la source de votre programme et si la frappe des SPC vous ennuie, il suffit de laisser des lignes vides aux endroits qui vous conviennent, cela aura le même effet au niveau du listing.

— NAM ou TTL : permet de donner un titre dont la longueur peut atteindre 32 caractères à un listing. Si l'option pagination a été choisie, ce titre est imprimé automatiquement en haut de chaque page. Le titre ainsi défini n'a aucun rapport avec le nom du fichier assemblé et vous pouvez faire ce que vous voulez à ce niveau. L'utilisation est élémentaire puisqu'il suffit d'écrire TTL (ou NAM) TEXTE, où TEXTE représente le titre

choisi. Il est possible d'avoir autant de directives TTL que vous le souhaitez dans un même programme (pour donner des noms différents à des sections diverses d'un même programme, par exemple).

— STTL : permet de définir un sous-titre qui sera imprimé en dessous du titre défini par un TTL en haut de chaque page si l'option pagination a été choisie. Ce sous-titre peut avoir jusqu'à 32 caractères, et autant de directives STTL que vous le souhaitez peuvent apparaître dans un même programme. L'utilisation est identique à celle de TTL ou NAM, puisqu'il suffit d'écrire : STTL TEXTE. Pour désactiver cette fonction en cours de listing, il suffit de placer un STTL sans texte.

— ERR : permet de faire afficher des messages d'erreur qui vous sont propres, c'est-à-dire des messages qui ne sont pas générés par l'assembleur mais par vous-même. Bien que cela vous paraisse étrange, c'est une pratique très utile en assemblage conditionnel pour savoir ce qui se passe, comme nous le verrons dans la suite de ce mode d'emploi. L'utilisation est la suivante : ERR TEXTE, où TEXTE est le message qui sera imprimé lorsque l'assembleur passera sur la directive ERR. Le fait que l'assembleur passe sur cette directive est pris en compte au niveau du nombre total d'erreurs détectées qui est affiché en fin de listing.

— RPT : permet d'indiquer à l'assembleur que la ligne qui suit doit être répétée N fois. Cela s'utilise de la façon suivante :

RPT N
LIGNE A REPETER
où N spécifie le nombre de répétitions de la ligne. Ainsi,

lorsque l'on fait des décalages multiples, n'est-il pas nécessaire de frapper autant d'ASL ou ASR que nécessaire ; il suffit de faire RPT N puis une seule fois ASR ou ASL, ils seront répétés N fois. D'autres utilisations sont possibles, l'assembleur ne se souciant pas du contenu de la ligne à répéter. Certaines directives telles que IF ou MACRO sont impossibles à répéter car cela est illogique ; si vous le demandez quand même, la directive RPT est ignorée.

— LIB : permet d'appeler, au sein de votre programme source, un fichier disque quelconque. L'utilisation en est la suivante : LIB NOM DE FICHIER, où NOM DE FICHIER est le nom du fichier que vous voulez voir inclure dans votre listing à l'emplacement du LIB. L'extension par défaut est TXT, et le lecteur par défaut est celui de travail. Cette directive est très utile pour des sous-programmes que vous utilisez souvent dans vos programmes. Ainsi, si vous avez, par exemple, un sous-programme de sortie de caractères sur le terminal, il vous suffit de le frapper une fois pour toutes et de le placer dans un fichier. Dès que vous en aurez besoin dans un programme et à l'emplacement voulu, il vous suffira de faire un LIB avec ce nom de fichier, il peut y avoir autant de LIB que vous le souhaitez dans un programme et les LIB peuvent même être imbriqués les uns dans les autres (c'est-à-dire qu'un LIB peut en appeler un autre) jusqu'à 12 niveaux d'imbrication.

— OPT : permet de spécifier un certain nombre d'options en plus de celles que vous avez pu définir sur la ligne d'appel de l'assembleur. Ces options sont résumées dans le tableau

de la figure 8 et s'utilisent de la façon suivante : OPT OPTION1, OPTION2..., OPTIONN. L'option PAG permet la pagination du listing avec numérotation des pages et impression des titres et sous-titres éventuels ainsi que de la date en haut de chaque page ; son utilisation est fortement conseillée pour avoir des listings propres et faciles à classer. L'option CON permet d'imprimer le code d'assemblage conditionnel qui a été sauté. MAC fait imprimer les lignes d'appel des macro-instructions. EXP fait imprimer les macro-instructions en mode expansé (voir la suite du mode d'emploi pour ces définitions). Le tableau de la figure 8 précise aussi quelles options sont prises par défaut et quelle est le contraire des options présentées ci-avant (NOE pour contraire de EXP, etc.).

Nous en avons terminé avec les directives d'assemblage dites classiques. Nous allons maintenant voir ce que permet ce macro-assembleur au niveau assemblage conditionnel et macro-instructions. Certains concepts exposés ci-après vont peut-être vous sembler étranges si vous êtes néophyte en programmation assembleur, ne vous en inquiétez pas, il vous suffira de ne pas faire appel à ces possibilités au début ; par la suite, lorsque vous serez devenu un programmeur chevronné, vous apprécierez leur existence.

L'assemblage conditionnel

Supposons que vous ayez un programme susceptible de tourner sur plusieurs systèmes différents, mais que chaque système nécessite des sous-

Nom de l'option	Contraire	Fonction de l'option	Par défaut
PAG	NOP	Pagination du listing	NOP
CON	NOC	Impression code conditionnel non assemblé	NOC
MAC	NOM	Impression des lignes d'appel des macros	MAC
EXP	NOE	Impression des macros en mode expansé	NOE

Fig. 8. — Tableau des options, de leurs fonctions et du choix pris par défaut par l'assembleur.

programmes d'entrée/sortie différents. Plutôt que de préparer quatre fichiers source de ce programme avec, pour chacun, les sous-programmes correspondants, il est possible, grâce à l'assemblage conditionnel, de n'avoir qu'un fichier source contenant tous les sous-programmes possibles et, par changement d'un seul paramètre au sein d'une directive EQU, par exemple, de faire assembler le programme correspondant au système désiré. Cette possibilité est extrêmement intéressante car elle allège votre stock de fichiers de façon importante ; de plus, elle vous permet d'avoir immédiatement un programme pour le système désiré en ne faisant qu'une édition très rapide au niveau du source général. Le principe en est fort simple. Au moyen de directives adéquates, on indique à l'assembleur qu'il doit calculer une expression (expression qui est fonction du paramètre que nous avons évoqué ci-avant) et, selon le résultat de ce calcul, il doit assembler ou non tel ou tel morceau de programme. Deux directives sont principalement utilisées : IF et ENDIF. Leur utilisation se fait comme indiqué figure 9. L'expression qui suit le IF est calculée. Si elle est vraie, c'est-à-dire si elle donne un résultat non nul, les lignes comprises entre IF et

ENDIF sont assemblées ; si l'expression est fautive, c'est-à-dire si elle donne un résultat nul, les lignes comprises entre IF et ENDIF sont ignorées.

Pour simplifier encore la manipulation de cet assemblage conditionnel, une troisième directive a été introduite : c'est le ELSE qui a quasiment le même sens que dans le IF THEN ELSE du Basic. La figure 10 vous indique comment placer ces directives, et ce qui se passe alors.

Il est possible d'imbriquer les « boucles » IF-ELSE-ENDIF les unes dans les autres, mais il faut prendre la précaution de faire des imbrications analogues à des cercles concentriques, c'est-à-dire qu'il ne faut pas sortir d'une boucle dont le ENDIF n'a pas été atteint.

Pour simplifier encore votre travail, il est possible de faire appel à la directive IFN, qui fonctionne comme le IF mais qui inverse le sens de la condition, comme le montre la figure 11.

Enfin, il existe encore une troisième forme de la directive IF qui est le IFC, présenté figure 12. La décision d'assemblage conditionnel n'a plus lieu en fonction de la valeur d'une expression, mais si les deux chaînes de caractères sont identiques ou non. Ces chaînes doivent être spécifiées de la façon suivante :

— Elles doivent être séparées par une virgule sans aucun espace.

— Chaque chaîne doit être comprise entre deux délimiteurs qui peuvent être soit l'apostrophe (') soit le guillemet (« ») ou...

— Chaque chaîne doit être constituée par une suite de caractères sans espace intercalaire. Ainsi ABCD sera une chaîne valide, mais AB CD ne le sera pas, et il faudra alors le noter « AB CD » ou 'AB CD'.

— Une chaîne vide peut être spécifiée en mettant deux délimiteurs côte à côte sans caractère entre eux.

Cette directive IFC existe aussi en version IFNC, et les portions de codes assemblées, indiquées figure 12, sont alors inversées (de même que le IFN inversait la figure relative au IF).

Enfin, dernière forme de cet assemblage conditionnel, le IF SKIP. Cette forme n'utilise plus qu'un IF tout seul, sans ENDIF pour terminer la zone conditionnelle. Le IF s'utilise alors de la façon suivante : IF EXPRESSION, N ou IFC CHAINE1, CHAINE2, N, où N est un nombre décimal compris entre 1 et 255 qui peut être précédé d'un signe + ou - et qui indique le nombre d'instructions à sauter (après cette ligne, dans le cas d'un signe +, ou avant, dans le cas

d'un signe -) si l'expression est fautive.

Attention ! ce mode de travail très particulier n'est autorisé qu'au sein de macro-instructions et non dans les parties « conventionnelles » d'un programme ; toute utilisation incorrecte donne lieu à un message d'erreur. Il est bien sûr possible d'employer cette forme avec IFN ou IFNC ; dans ce cas, les lignes sont sautées si l'expression ou la comparaison est vraie.

Une seule précaution est nécessaire lors de l'emploi de l'assemblage conditionnel : il faut que toutes les étiquettes qui apparaissent dans les expressions utilisées pour la décision de l'assemblage conditionnel aient été définies au préalable.

Les macro-instructions

Cette possibilité, qui n'existe que sur les macro-assembleurs, offre une souplesse et une puissance de programmation considérable, comme vous pourrez vous en rendre compte à l'usage. Une macro-instruction n'est rien d'autre qu'un ensemble d'instructions (6809, dans notre cas) que vous pouvez ensuite appeler dans le programme par un seul nom qui est le nom de la macro-instruction. En d'autres termes, avec un macro-assembleur, vous pouvez créer des mnémoniques nouveaux et des « instructions » nouvelles. Cette façon de faire vous montre clairement que l'utilisation de macro-instructions passe par deux phases : une phase de définition ou de création de la macro, pendant laquelle vous lui donnez son nom et vous indiquez la liste des instructions qu'elle va remplacer, et une phase d'utilisation proprement dite où vous placerez son nom aux endroits désirés dans votre programme et où, lorsqu'il arrivera à ces endroits, l'assembleur opérera automatiquement la substitution du nom par la suite d'instructions que vous avez définie au préalable. Grossièrement, et si l'on s'en tient à cette présentation, on peut assimiler une macro-instruction à un sous-programme qui, au lieu

```
IF EXPRESSION
-
- Lignes assemblées si expression vraie
-
ENDIF
```

Fig. 9. — Utilisation IF-ENDIF

```
IF EXPRESSION
-
- Assemble si expression vraie
-
ELSE
-
- Assemble si expression fautive
-
ENDIF
```

Fig. 10. — Utilisation du IF-ELSE-ENDIF.

```
IFN EXPRESSION
-
- Assemble si expression fautive
-
ELSE
-
- Assemble si expression vraie
-
ENDIF
```

Fig. 11. — Utilisation du IFN

```
IFC CHAINE1, CHAINE2
-
- Assemble si CHAINE1 = CHAINE2
-
ELSE
-
- Assemble si CHAINE1 ≠ CHAINE2
-
ENDIF
```

Fig. 12. — Utilisation du IFC

d'être appelé par un BSR ou LBSR chaque fois que c'est nécessaire, serait tout simplement ré-écrit dans le programme principal. Nous allons voir qu'en fait cela va beaucoup plus loin, mais ne brûlons pas les étapes.

Nous allons prendre en exemple une macro-instruction simple, comme indiqué figure 13. La succession d'instructions que nous avons écrites permet de réaliser un décalage arithmétique à gauche de quatre positions du registre D (double accumulateur formé par la mise bout à bout de A et B). Pour faire une macro-instruction de cela, il suffit d'écrire, comme nous l'avons fait, ASLD4 MACRO en début de cette suite d'instructions pour que l'assembleur, voyant le mot clé MACRO, affecte l'étiquette qui précède à la macro-instruction constituée par toutes les instructions comprises entre le mot MACRO et le mot ENDM. ASLD 4 est le nom de la macro-instruction, la suite des instructions est le corps de la macro ou son expansion.

Une telle définition doit être faite pour toutes les macro-instructions que vous voulez utiliser dans un programme (le nombre de macro n'est limité que par la taille de votre mémoire), et ces définitions doivent intervenir avant toute utilisation de la macro concernée; une bonne habitude consiste à les placer en début de programme, pour être tranquille.

La figure 14 vous montre alors comment l'on appelle la macro dans le listing source et ce qu'en fait l'assembleur au niveau du listing définitif du programme. Nous constatons qu'il remplace bel et bien la macro par la liste d'instructions équivalentes.

Si les macro-instructions ne se limitaient qu'à cela, leur intérêt serait assez limité; mais il y a mieux, lorsque l'on aborde la possibilité de passage de paramètres. Soit, par exemple, le petit programme de la figure 15 qui a pour fonction d'ajouter NUM1, NUM2 et NUM3 et de mettre le total dans SOMME. Bien que ce programme dépende de valeurs qui lui sont « extérieures »

*EXEMPLE DE MACRO INSTRUCTION

```
ASLD4  MACRO
        ASLB
        ROLA
        ASLB
        ROLA
        ASLB
        ROLA
        ASLB
        ROLA
        ENDM
```

Fig. 13. — Exemple de macro-instruction.

*ADDITION DE TROIS NOMBRES

```
000B NUM1 EQU *
000B NUM2 EQU *
000B NUM3 EQU *
000B SOMME EQU *

000B DC 0B ADD3 LDD NUM1
000A D3 0B ADDD NUM2
000C D3 0B ADDD NUM3
000E DD 0B STD  SOMME
```

Fig. 15. — Exemple de programme d'addition des trois nombres NUM1, NUM2, et NUM3.

*APPEL DE LA MACRO INSTRUCTION DANS UN PROGRAMME

```
0000 ASLD4
0000 5B ASLB
0001 4F ROLA
0002 5B ASLB
0003 4F ROLA
0004 5B ASLB
0005 4F ROLA
0006 5B ASLB
0007 4F ROLA
ENDM
```

Fig. 14. — Exemple d'assemblage et d'appel de la macro de la figure 13.

*EXEMPLE DE MACRO INSTRUCTION AVEC PARAMETRES

```
ADD3  MACRO
        LDD  &1
        ADDD &2
        ADDD &3
        STD  SOMME
        ENDM
```

Fig. 16. — Réalisation avec une macro-instruction du programme de la figure 15.

*APPEL DE MACRO AVEC PARAMETRES

```
0000 ADD3 NUM1, NUM2, NUM3
0000 DC 02 LDD NUM1
0002 D3 04 ADDD NUM2
0004 D3 06 ADDD NUM3
0006 DD 00 STD SOMME
ENDM
```

Fig. 17. — Appel de la macro de la figure 16 avec passage de paramètres.

*LES PARAMETRES PEUVENT TOUT REMPLACER

```
DEMO  MACRO
        LDA  $&1
        LDB  E&1
        NOP
        NOP
        &3
        TST  M&1M
        ENDM
        LES PARAMETRES VONT MEME
        DANS LES &2
```

*APPEL DE LA MACRO PRECEDENTE

```
000B DEMO 1000, "COMMENTAIRES", 'LDA #3', TOTO
000B B6 1000 LDA $1000
000B D6 0B LDB E1000
000D 12 NOP
000E 12 NOP
000F 86 03 LDA #3
0011 0D 0A TOTO TST M1000M
ENDM
        LES PARAMETRES VONT MEME
        DANS LES COMMENTAIRES
```

Fig. 18. — Exemple montrant la polyvalence des paramètres dans une macro-instruction.

(NUM1, NUM2 et NUM3), il est possible d'en faire une macro-instruction, visible en figure 16. Nous y remarquons que les trois nombres ont été remplacés par le symbole « et commercial » (&), suivi par un chiffre de 1 à 3. Lorsque nous allons appeler cette macro dans notre programme, nous allons le faire de la façon suivante : ADD3 NUM1, NUM2, NUM3 ; cela aura pour effet de faire substituer à l'assembleur &1 par NUM1, &2 par NUM2 et &3 par NUM3, et il produira le listing de la figure 17.

En d'autres termes, il est possible de définir dans une macro jusqu'à neuf paramètres qui seront impérativement représentés par un « et commercial » (&), suivi par un chiffre de 1 à 9. Lors de l'appel de la macro, la ligne d'appel devra contenir autant de paramètres que nécessaire pour la macro concernée, et le premier paramètre remplacera &1, le deuxième &2, etc., le neuvième &9. Si un nombre insuffisant de paramètres est fourni sur une ligne d'appel de

macro, tous les &N non satisfaits seront remplacés par des zéros.

Les paramètres passés doivent être fournis comme indiqué ci-avant, séparés par un espace du nom de la macro et séparés entre eux par des virgules. Ils peuvent être des constantes numériques mais aussi des étiquettes, des chaînes de caractères, ou même des instructions. Les chaînes de caractères doivent être comprises entre deux guillemets (« ») ou deux apostrophes (' '), ou être constituées par une suite de caractères alphanumériques sans espace. La figure 18 montre un exemple bien choisi qui met en application le fait que les paramètres peuvent être à peu près n'importe quoi. Remarquez qu'un même symbole &N peut apparaître plusieurs fois dans une macro ; il sera à chaque fois remplacé par la bonne valeur, c'est-à-dire celle qui correspond à la N° position dans la ligne d'appel de la macro.

Pour accroître encore la souplesse d'utilisation des

macro-instructions, deux possibilités vous sont encore offertes. La première est celle de la directive EXITM. Cette directive permet de ne pas générer tout le corps de la macro, mais au contraire d'en sortir prématurément. Cela peut sembler illogique à première vue, mais ce n'est pas le cas si vous vous remémorez les possibilités d'assemblage conditionnel vues ci-avant. La figure 19 vous montre un exemple d'utilisation de cette directive ainsi que deux lignes d'appel de cette macro qui conduiront ou non à sa génération complète.

La dernière possibilité est celle offerte par les directives DUP en ENDD. Ces deux directives permettent de dupliquer autant de fois que vous le désirez (mais pas au-delà de 255 fois) un certain nombre de lignes d'instructions. La figure 20 vous donne le principe d'emploi de DUP et un exemple d'utilisation. Remarquez que l'intérêt principal de cette directive réside dans le fait que le nombre de duplications puisse être un paramètre. A propos de ce nombre, il est toujours placé après DUP dont il est séparé par un espace, et peut être constitué par une expression dont la valeur doit être comprise entre 1 et 255. Les « boucles » DUP ENDD ne doivent pas être enchevêtrées (c'est-à-dire qu'il ne doit pas y avoir de DUP ENDD dans une « boucle » DUP ENDD).

quette et être suivi d'opérandes (qui sont les paramètres passés à la macro).

— Une macro doit toujours être définie avant d'être appelée ; d'où notre conseil, déjà donné, de définir les macros en début de programme.

— Les macros peuvent être enchevêtrées lors de leur appel, mais aussi lors de leur définition ; c'est-à-dire qu'une macro peut faire appel ou référence à d'autres macros.

— Les lignes de commentaires sont effacées des macro-instructions par l'assembleur pour économiser la place mémoire.

— Les étiquettes locales au sein d'une macro ne sont pas autorisées ; en effet, à chaque appel de la macro concernée, il y aurait génération de la même étiquette, et cela conduirait l'assembleur à générer un message d'erreur de définition multiple d'un symbole.

— Les directives LIB sont interdites au sein des macro-instructions.

— Une fois qu'une macro a été définie dans un programme, elle ne peut plus être annulée ni redéfinie.

— La table des noms de macro est explorée par l'assembleur avant la table des instructions du 6809, ce qui signifie que vous pouvez remplacer un mnémonique 6809 par la macro de votre choix sans créer d'erreur.

— Un contrôle du nombre de paramètres demandés et du nombre de paramètres fournis lors de l'appel d'une macro n'est pas réalisé par l'assembleur, ce qui signifie que tous les paramètres non pourvus seront remplacés par des zéros sans que cela génère de message d'erreur sur le listing.

ABCD MACRO

```

-
-           Toujours généré
-
IFNC &2, OUI
EXITM
-
-           Généré seulement si &2 = OUI
-
ENDM
    
```

ABCD 'PARAMETRE1',OUI fera générer toute la macro
 ABCD 'PARAMETRE1',NON fera générer la première partie de la macro

Fig. 19. — Exemple d'utilisation de la directive EXITM

```

*FONCTION DE DUP
ASLDN  MACRO
        DUP    &1
        ASLB
        ROLA
        ENDD
        ENDM

*APPEL DE ASLDN AVEC N = 3
0000    ASLDN  3
0000 58    ASLB
0001 49    ROLA
0002 58    ASLB
0003 49    ROLA
0004 58    ASLB
0005 49    ROLA
        ENDM
    
```

Fig. 20. — Exemple d'utilisation de la directive DUP.

Remarques et restrictions relatives aux macro-instructions

Les macro-instructions ne sont pas difficiles d'emploi une fois que l'on a osé essayer ; il faut cependant respecter quelques règles principales rappelées ci-dessous :

— Tout d'abord, le nom d'une macro joue un double rôle ; celui d'étiquette lors de la définition de la macro, puisque l'on écrit NOM MACRO et c'est MACRO qui est la directive d'assemblage. Ensuite, ce nom joue le rôle d'une instruction toutes les fois que l'on appelle la macro ; il peut même être précédé d'une éti-

Quelques informations

Cet article étant fort long, nous allons être brefs ; quelques photos ci-jointes vous montrent le boîtier réalisé par Saint-Ignan Informatique pour notre mini-ordinateur ; précisons que ce boîtier reçoit les cartes au format « Facim » et au format « Exorciser-Motrola ». Nous vous en parlerons avec plus de détail le mois prochain.

Nous avons oublié de vous

Adresse	Contenu
00 à DF	F
EO	0
E1 à EA	2
EB à FF	F

Fig. 21. — Contenu de la PROM DECFL09

donner le contenu de DEC-FLOP09 ; cette erreur est réparée figure 21.

Nous présentons nos excuses aux personnes qui ont un peu attendu leurs lettres d'informations 6809 et leurs programmes début 1983 ; le déménagement de l'auteur a en effet allongé quelque peu ses délais de réponse à cette période.

Pour répondre à un problème qui nous est très souvent soumis, nous consacrons quelques lignes, dans notre prochain numéro, au choix d'une imprimante.

Toujours pour répondre à des questions qui reviennent également souvent, l'auteur précise que, hormis les composants électroniques classiques de cette réalisation qui peuvent être approvisionnés quasiment partout (Pentasonic par exemple les possède quasiment tous), les composants particuliers tels que circuits imprimés, transformateur, boîtier, mémoires de décodage d'adresse pré-programmées, ne sont fournis à sa connaissance et avec son autorisation que par trois sociétés qui sont

Facim, Saint-Ignan Informatique et Micropross. Il décline en particulier toute responsabilité quant au mauvais fonctionnement du système dû à un approvisionnement de ces composants particuliers (surtout pour les mémoires pré-programmées) ailleurs, et ce jusqu'à plus ample informé. De même, et pour les lecteurs de disques souples, malgré de belles promesses faites par divers fournisseurs, nous n'avons essayé que des Tandon et des MPI pour l'instant, et nous continuons à ne préconiser que ces modèles. Les annonceurs garantissant la compatibilité de « leurs » lecteurs avec notre système le font sous leur seule et entière responsabilité.

Conclusion

Cet article vous aura peut-être semblé un peu lourd à digérer, surtout si vous n'êtes pas un passionné d'assembleur ; il était cependant nécessaire pour vous présenter complètement ce programme dont les possibilités, une fois que l'on sait les exploiter, sont immenses. Si certains points vous semblent obscurs, n'oubliez pas notre conseil maintes fois répété : essayez et vous verrez...

C. TAVERNIER
(A suivre.)

Bloc-notes

BIBLIOGRAPHIE

Initiation — Business Basic par Eddie Adams

Ce livre explore progressivement et complètement le langage Business Basic de l'Apple III, plus particulièrement orienté vers les applications de gestion.

Les caractéristiques essentielles de ce langage y sont mises en évidence ainsi que les règles et les concepts généraux et fondamentaux, afin de permettre au lecteur :

— de comprendre et d'assimiler aisément le principe de fonctionnement de chaque ins-

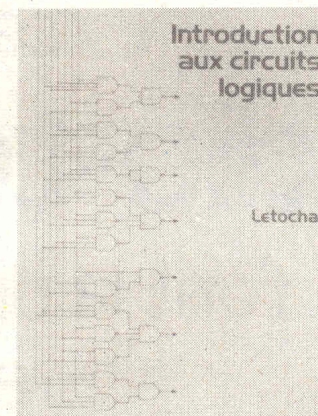
truction, commande, opérateur ou symbole ;

— d'en utiliser toutes les ressources dans des programmes qu'il aura pu concevoir et créer lui-même de façon progressive, à partir d'exemples donnés et de notions de base largement commentées et explicitées.

Table des matières : Les nombres — Les variables — Les fonctions — Les fonctions arithmétiques — Les fonctions trigonométriques — Les fonctions de conversion de type — Les définitions de fonction — Les chaînes — Les fonctions de chaîne — Les instructions — Les expressions — Les expressions arithmétiques — Les expressions alphanumériques — Les expressions logiques — Les instructions d'affectation — Les sous-programmes — Les transferts de contrôle — La mise au point — La correction programmée des erreurs — Les boucles — Listes et tableaux — La lecture des données — L'exécution d'un programme — L'affichage à l'écran — La définition des zones d'impression — Les fichiers — Index des mots réservés.

Editeur McGraw-Hill. En vente à la Librairie Parisienne de la Radio.

Introduction aux circuits logiques par J. Letocha



Ce volume d'introduction aux circuits logiques est destiné principalement aux étudiants en électrotechnique. Chaque chapitre contient une série d'objectifs, un texte d'introduction, un cours théorique, et se termine par un ensemble de problèmes. A l'intérieur de chaque chapitre, des sous-sections indépendantes traitent chacune d'un sujet bien défini. La partie cours est illustrée par de nombreux exemples et schémas. En appendice sont données les solutions à de nombreux problèmes proposés en fin de chapitres.

Table des matières : Systèmes de numération : Base

d'un système de numération. Changement de base. Opérations arithmétiques en binaire. La complémentation. Codes. Codage. **Algèbre de Boole** : Les opérations ou fonctions de base de l'algèbre de Boole. Application à un réseau électrique. Axiomes ou lois fondamentales de l'algèbre de Boole. Evaluation d'une fonction logique. Table des fonctions de deux variables. Relations de base de l'algèbre de Boole. Théorèmes de De Morgan. Dualité de l'algèbre de Boole. Simplification algébrique des équations booléennes. **Représentation, simplification, implantation des fonctions logiques** : Modes de représentation des fonctions logiques. Formes canoniques. Simplification par la table de Karnaugh. Quelques circuits intégrés d'implantation d'une fonction logique. Implantation d'une fonction logique à grand nombre de variables. **Problèmes de logique combinatoire. Logique séquentielle** : Circuits synchrones et circuits asynchrones. La bascule JK. Etude des bascules en circuits intégrés. Applications des bascules. Les compteurs. Les registres à décalage. Appendice. Lexique. Index.

Editeur : McGraw Hill.

