ODIN

OUTIL DE DEVELOPPEMENT INTERACTIF POUR THOMSON MO5

Assembleur Symbolique
Desassembleur
Moniteur et Editeur pleine page

Permettez-nous de vous féliciter; vous venez d'acquérir l'un des outils de développement des plus complets, disponible pour Micro-Ordinateur.

Avec ODIN, vous allez pouvoir utiliser toutes les potentialités de votre MO 5.

Attention: ODIN est un outil au nombre impressionnant de fonctionnalités, il vous faudra compter sur une période d'apprentissage, surtout si vous n'avez jamais programmé en langage machine.

Ce Manuel ne remplace absolument pas un livre sur la programmation du microprocesseur 6809, ouvrage qui vous sera nécessaire de toute façon.

Afin de répondre aux nombreuses questions que vous êtes en droit de vous poser concernant la programmation du MO 5 en assembleur, nous avons écrit un livre édité aux éditions du P.S.I., intitulé "Assembleur et périphériques des MO5 et TO7-70".

Outil de Développement Interactif sur 6809

DESCRIPTION DU SYSTEME

ODIN 6809 constitue un véritable outil de programmation du micro-processeur 6809E qui équipe votre MO5. Il est composé de 3 modules:

Le premier est un Moniteur qui intègre parmi ses 26 commmandes:
– Un DESASSEMBLEUR 6809E symbolique ou non, selon votre choix.
– Les classiques "DUMP" mémoire en représentations ASCII ou Binaire.
– Des fonctions rapides de RECHERCHE et de REMPLACEMENT d'octets spécifiés sous n'importe quelle forme. (Ascii, Hexa, Binaire, Déci).
– La gestion complète de POINTS D'ARRET pour le contrôle d'exécution des programmes-machine.
– Tous les modes courants de REMPLISSAGE et de DEPLACEMENT de blocs mémoire.
– Une commande d'exécution PAS-A-PAS d'un programme machine, même situé en ROM, pour une mise au point facile.
– Un CALCULATEUR qui évalue des expressions complexes, manipule des chaînes binaires, utilise les fonctions logiques grâce à ses 17 opérateurs disponibles.
– Une commande de RELOCATION UTILISATEUR qui vous permettra de reloger tous vos programmes-machine.
– Enfin, une fonction de RELOCATION SYSTEME pour installer ODIN dans une zone de la RAM à votre convenance.

ODIN, c'est aussi un module EDITEUR de textes source destinés à être traduits par l'Assembleur. Si le langage machine ne vous est pas encore familier, vous pouvez à juste titre vous interroger sur la nature de ce langage, et sur l'intérêt de son utilisation.
Regardez donc le programme BASIC suivant dont le but est de déplacer vers la droite la fenêtre de texte du MO5, puis son équivalent en langage d'assemblage:

```
10 FOR I=1 TO 40 ..................................  40 décalages
20 FOR J=0 TO 199 .................................  200 rangées à décaler
30 FOR K=39 TO 0 STEP-1 ...........................  sur 40 colonnes
40 POKE (J* 40+K), PEEK (J* 40+K-1) ...............
45 NEXT K: POKE J*40,0 ............................  Efface la colonne 0
50 NEXT J: NEXT I .................................  et boucle...
```

Nous avons interrompu ce programme avant terme, au bord de la crise de nerfs, après 2mn20 de supplice pour le MO5, temps d'un seul décalage!

Maintenant, voici son équivalent en langage d'assemblage:

```
00010          LDA # $28
00012          * 40 Décalages
00015          INIT CLRB
00020          PAGE PSHS A,B
00030               LDA # 40
00040               MUL
00050               TFR D,X
00055          * Calcul de la ligne courante -> X
00060               LDA # 38
00065          * 40 colonnes à décaler sur 1 lgn
00070          COL  LDB A,X
00080               INCA
00090               STB A,X
```

```
00100               DECA
00110               DECA
00120               BPL COL
00122          * Exécution du décalage d'1 lgn
00125               CLR   ,X
00127          * Effacement colonne 0 de la lgn
00130               PULS A,B
00140               INCB
00150               CMPB #199
00160               BCS PAGE
00165          * Page complète (200 l.) décalée ?
00170               DECA
00180               BNE INIT
00185          * Page décalée 40 fois ?
00190               RTS
```

Ce programme quant à lui, s'exécute en moins de 7 secondes...

Maîtriser le langage d'assemblage, c'est surtout être à même d'exploiter pleinement les capacités du processeur 6809E qui équipe votre MO5. Ainsi, vous contrôlerez le cœur de votre machine et vous aurez à votre disposition les multiples routines incorporées à la ROM qu'ODIN vous aidera à détailler et à comprendre.

Pour aboutir au programme objet qui a opéré le décalage d'écran, il vous faudra auparavant être passé par la phase d'édition d'un texte Source qui servira de référence au travail de traduction en code 6809E de l'Assembleur. Or le module EDITEUR est résolument destiné à faciliter cette étape, parce que les concepteurs d'ODIN savent combien un bon éditeur hâte la mise au point des programmes. Les caractéristiques de cet éditeur sont entre-autres :

– Un mode d'édition du texte Source, dérivé de la pleine-page, qui permet d'accéder très facilement à n'importe quelle zone du texte. De plus, il assure le CONTROLE de l'entrée de LIGNES source de façon à vous prévenir IMMEDIATEMENT d'une éventuelle erreur de syntaxe dans le mnémonique ou dans l'étiquette. Ainsi, les risques d'erreurs bénignes - mais qui obligent à ré-assembler un Source parfois long - seront diminués d'autant.
– Des fonctions de MANIPULATION du texte Source, comme la renumérotation, la suppression, la copie et le transfert de blocs de lignes.
– Des commandes gérant les opérations CASSETTE pour la sauvegarde et la récupération du Source, y compris la fusion de fichiers selon deux critères possibles.
– Des fonctions spéciales très pratiques telles que la TRANSFORMATION de CODE-OBJET en SOURCE.
– La possibilité de faire des RECHERCHES et des REMPLACEMENTS partiels ou globaux de chaînes de caractères sur le texte Source en mémoire.
– Enfin, ODIN vous permettra de SAUVEGARDER, RECUPERER et REUTILISER à volonté les TABLES DE SYMBOLES constituées lors d'assemblages successifs avec des textes Source différents.

ODIN permet également d'accéder à BASIC à tout moment lorsque vous vous trouvez sous Moniteur ou sous Editeur. Il vous sera bien entendu possible de réintégrer ODIN par le simple envoi d'un mot-clé, sans aucune perte du Source éventuellement en mémoire.

Le troisième module composant ODIN 6809 est un ASSEMBLEUR SYMBOLIQUE COMPLET qui offre :
– L'assemblage avec co-résidence de l'objet en mémoire, ou avec écriture directe de ce code sur cassette.
– L'assemblage CONDITIONNEL.
– La récapitulation finale des éventuelles erreurs rencontrées au cours de l'assemblage.

PRESENTATION GENERALE DU MODE DE COMMANDE

L'affichage

Lorsque vous entrez sous Moniteur comme sous Editeur, l'écran se divise en deux zones distinctes.

L'une est située sur la première ligne de l'écran et réservée au système. Elle comprend les indications "Source:" et "Mem:" inscrites en noir sur fond blanc. Leurs significations seront abordées plus avant dans la section "DESCRIPTION DE L'ASSEMBLEUR".

L'autre zone, c'est-à-dire le reste de l'écran, vous est accessible pour entrer des commandes. A la mise en route du système, elle est orange sous Moniteur et noire sous Editeur.

Lorsqu'il sera prêt à accepter vos commandes, le système affichera l'un des deux caractères "*" ou "&" accompagné du curseur. Ce caractère vous indiquera en outre si vous vous trouvez respectivement sous Moniteur ou sous Editeur.

L'entrée-ligne

Chacune des 50 commandes du système complet est représentée par un symbole alphanumérique. Elle peut selon sa syntaxe être suivie d'un certain nombre de paramètres. Notez dès à présent qu'un même symbole pourra parfois correspondre à deux ordres totalement différents selon qu'il aura été utilisé sous Moniteur ou sous Editeur.

Nous avons conservé les facilités de l'entrée-ligne BASIC pour le mode commande commun au Moniteur et à l'Editeur. Ainsi, vous disposez toujours de l'édition "pleine page" et de l'entrée-ligne à commandes multiples.

Il vous suffira pour utiliser cette dernière de séparer chaque ordre distinct par le caractère ":", également reconnu de BASIC. Cependant, certaines commandes empêchent l'exécution de tout ordre suivant sur la ligne. Celui-ci sera simplement ignoré par l'interpréteur.

Ces commandes particulières sont:

– pour le Moniteur: C, J, K, Q, S, U, W, X, /

– pour l'Editeur: A, E, K, M, Q, W, et /

Le contrôle d'exécution

Les commandes d'ODIN qui affichent pendant leur traitement le contenu de la mémoire disposent des fonctions de Pause et d'Arrêt d'exécution. Ainsi, il vous est possible de geler momentanément le déroulement et donc l'affichage d'un ordre en pressant la barre d'espacement. Tout appui sur une touche différente exceptée «STOP» le fera repartir. Si vous choisissez d'appuyer de nouveau sur la barre d'espacement, vous entrerez alors dans un mode d'exécution ligne à ligne, qui se poursuivra à chaque nouvelle pression sur cette barre. Si vous appuyez sur «STOP», et quelque soit l'état de votre affichage, vous arrêterez définitivement la commande jusqu'alors exécutée.

La représentation des valeurs numériques

Sous Moniteur:

En règle générale, le Moniteur affichera toujours ses valeurs et autres adresses en hexadécimal. Les quelques exceptions existantes (<L> et <@>) vous seront décrites plus avant.

En ce qui concerne l'entrée de ces données numériques, vous pourrez la faire dans les trois bases suivantes:

Décimal, précédé de "&"
Binaire, précédé de "%"
Hexa, précédé de "$"

Cependant, l'Hexa étant la base par défaut sous Moniteur, le symbole "$" est presque toujours facultatif (sauf pour <@>). Notez encore que les chiffres composant un nombre binaire seront systématiquement justifiés à droite pour leur évaluation.

Sous Editeur:

Contrairement au Moniteur, le module Editeur ne traite pas des adresses mémoire mais des N^os^ de lignes du texte Source. Ils doivent être entrés en décimal et sans préfixe "&".

Cependant, quelques-unes de ses commandes font appel à des adresses pour paramètres. Dans ce cas, utilisez les normes définies au paragraphe précédent pour les entrer.

Sous Assembleur:

L'affichage du code-objet lors d'un assemblage sera toujours effectué en hexadécimal.

L'entrée d'expressions numériques dans un champ d'opérande devra suivre la syntaxe définie au premier paragraphe, exception faite des valeurs hexa qui devront impérativement être précédées de "$" afin de les distinguer des étiquettes alphanumériques.

Nota: Les expressions intégrées au texte Source sont identiques en tous points à celles qu'évaluera la commande <@>. Reportez-vous à ce paragraphe pour en connaître toutes les caractéristiques. De plus, une étude détaillée des champs de données existant dans la syntaxe assembleur est faite à la section 'DESCRIPTION DE L'ASSEMBLEUR' au paragraphe 'Structure d'un fichier Source'.

LES DIFFERENTS TYPES DE PARAMETRES

Lorsque nous aborderons la description particulière de chaque commande, nous utiliserons des abbréviations pour représenter les différents types de paramètres susceptibles de lui être nécessaires. Le tableau suivant donne l'abbréviation, la nature et la syntaxe dans la commande de chacun de ces paramètres valides sous Moniteur et/ou sous Editeur.

Si vous devez en spécifier plusieurs pour une seule commande, séparez-les par des virgules.

Nota: Certaines commandes Moniteur conservent la dernière adresse mémoire qu'elles ont traitée avant d'être interrompues. Nous appellerons par la suite "Cont" cette adresse incrémentée de 1.

Abbréviation	*Nature*	*Abbréviation*	*Nature*
Zmem	Définit une zone mémoire sur laquelle portera la commande. Il existe 5 syntaxes différentes pour ce paramètre: *adr1-adr2* délimite la zone comprise entre les adresses 1 et 2. *–adr2* marque la zone comprise entre Cont et adr2. *adr1–* représente l'espace mémoire situé entre adr1 et la fin de la mémoire. *adr1* porte sur la mémoire située entre adr1 et adr2 qui sera calculée par l'interpréteur en ajoutant un nombre d'octets par défaut propre à chaque commande. Celui-ci sera précisé pour chacune d'elles dans les sections de description des commandes. Si vous ne donnez pas de paramètre de zone mémoire à une commande qui ne le considère pas comme obligatoire, l'interpréteur prendra adr1 égal à Cont et réagira de la même façon que pour la syntaxe précédente.	Incr	Incrément. Ce paramètre servira à certaines commandes Editeur pour créer ou modifier tout ou partie des N^{os} de lignes de votre Source. Il représentera l'écart entre les valeurs des N^{os} de deux lignes consécutives. Incr doit être compris entre 1 et 32767. Ce paramètre est toujours facultatif. S'il n'est pas précisé, le dernier incrément utilisé sera pris par défaut. Il vaut 10 à l'initialisation du système.
		FichO	Nom de fichier obligatoire pour les opérations sur cassette. Il correspond aux normes BASIC et doit être encadré par des guillemets.
		FichF	Nom de ficher facultatif. S'il n'est pas précisé, le nom et l'extension stockés après l'indicateur "Source" seront utilisés. FichO et FichF identifient des fichiers Sources.
		Obj	Le nom de fichier Objet correspond au FichO exception faite de la nature de son contenu qui est le code-machine généré par un assemblage à partir du Source présent en mémoire.
Bloc	Définit sous Editeur uniquement une zone du programme source en mémoire par des N^{os} de lignes décimaux compris entre 1 et 65535. Il existe là encore plusieurs syntaxes pour ce types de paramètre: *lgn1-lgn2* représente une zone comprise entre les N^{os} de lignes lgn1 et lgn2 inclus. *lgn1–* porte sur les lignes comprises entre lgn1 et la dernière ligne du texte source. *–lgn2* marque une zone allant de la première ligne du source à lgn2. *lgn1* ne représente que la ligne spécifiée. *lgn1.n* délimite la partie du source commençant en lgn1 et finissant n–1 ligne(s) après lgn1.	Chaîne	Chaîne de caractères alphanumériques. Elle doit être encadrée par des guillemets. Cependant, si la chaîne termine l'entrée-ligne, le dernier guillemet est facultatif. "ODIN, 1985 Loriciels"
		Data	Chaîne de données. Celle-ci peut être constituée d'octets, de Chaînes ou d'une combinaison des deux. La longueur de Data ne devra cependant pas excéder 128 octets. (Chaque donnée doit être séparée de la suivante par une virgule). "Mon", 49, &84, "eur"

LE MODE D'EDITION

L'écran y est divisé en deux zones distinctes :
– la première peut contenir 22 lignes et visualise la partie du texte que l'on veut éditer ;
– la seconde est constituée d'une seule ligne de couleur différente au bas de l'écran, qui permet l'édition proprement dite du texte.

Ces zones seront respectivement appelées zone 1 et zone 2 dans la suite de cet explicatif.

Structure d'une ligne
L'entrée des données sur une ligne Source type d'ODIN est le suivant :
– Champ N° de ligne (0<n<65536).

Mis en tête de la zone 2, il doit être entré en décimal.
– Champ Etiquette (6 caractères maximum) commençant toujours par une lettre.
– Champ Mnémonique (5 caractères maximum)
– Champ d'Opérande (19 caractères maximum)

A chaque validation, les contenus des champs 1, 2 et 3 seront contrôlés, si une erreur était détectée, le curseur de la zone 2 serait positionné en tête du champ refusé.
De plus, si la place mémoire restante se révélait insuffisante pour y stocker la dernière ligne validée, le message "Mémoire saturée" apparaîtrait au bas de l'écran. Dans ce cas, frappez une touche quelconque pour reprendre la main.

Nota : Des précisions exhaustives vous sont données quant au contenu des champs dans la section "DESCRIPTION DE L'ASSEMBLEUR".

Les commandes du mode d'édition

Stop	Provoque le retour au mode de commande de l'Editeur.
Ins	Insère 1 caractère blanc à la place du curseur dans le champ courant de la zone 2.
Del	Détruit le caractère pointé par le curseur de la zone 2. Cette commande - comme la précédente n'agit que sur le champ courant.
Flèche haute	Porte la flèche de la zone 1 sur la ligne précédente du texte visualisé.
Flèche basse	Porte cette flèche sur la ligne suivante.
Flèche gauche	Décale vers la gauche le curseur de la zone 2.
Flèche droite	Décale le curseur de la zone 2 vers la droite.
Cnt-X	Efface tous les caractères à partir du curseur jusqu'à la fin de la zone 2, sans limitation de champs.
Basic flèche gauche	Met le curseur de zone 2 en début de ligne.
Basic flèche droite	Met le curseur 2 en fin de zone.
Basic flèche haute	Met la flèche de zone 1 sur la première ligne du texte. Cette commande visualise les 22 lignes de tête du source.
Basic flèche basse	Met la flèche de zone 1 sur la première des 22 dernières lignes du source qui seront visualisées sur cette zone.
Basic Del	Efface seulement les caractères du champ courant de la zone 2 à partir du curseur.
Basic Ins	Transfère les caractères du champ courant sur le champ suivant. Cette commande est une insertion, elle décale donc le précédent contenu du champ destinataire.

Entrée	Valide la ligne présente en zone 2, et déclenche sa vérification puis son insertion dans le source.
Acc	Appelle en zone 2 la ligne pointée par la flèche clignotante de la zone 1. Il vous sera possible de la modifier à votre gré, puis de la valider par "entrée" pour opérer la substitution.
Cnt-D	Enlève du source la ligne pointée par la flèche de zone 1.
Merge	Visualise en zone 1 les 22 lignes précédentes du source.
Raz	Visualise les 22 lignes suivantes du source en zone 1.
Espace	Positionne le curseur de zone 2 au début du champ suivant. Cependant, la commande est inopérante lorsque le curseur est déjà sur le champ d'opérande. Dans ce cas, la touche Espace retrouve sa fonction normale en écrivant un blanc à l'endroit du curseur.

LES COMMANDES DU MONITEUR

La présence du Moniteur est signalée par le symbole "*" qui indique que le système est prêt à accepter vos commandes.

Cette section abordera cas par cas les différentes commandes disponibles sous Moniteur. Le nombre d'octets par défaut sur lesquels porteront éventuellement certaines commandes est signalé par l'indication: "Déf:".

<A>
Dans le but de vous faciliter la mise au point des programmes, le Moniteur est capable d'utiliser la table des symboles créée par le dernier assemblage. Ainsi la commande de désassemblage (L) remplacera à l'écran les adresses hexadécimales par leurs symboles associés. Il vous est cependant possible d'autoriser ou d'inhiber un tel remplacement. Pour ce faire, entrez simplement:

A+ pour permettre
ou A− afin d'empêcher

la symbolisation.
Pour plus de détails sur les modalités de cette fonction, reportez-vous au paragraphe <L> de cette même section.

<B> B Zmem Déf: 21
Cette commande affiche le contenu de la mémoire dans une représentation hexadécimale et binaire.
Les bits à 1 seront symbolisés par "*", les bits à 0 par "-".

Exemple: *B FCB7-FCB8

	=>	
FCB7-	28	--*-*---
FCB8-	FE	*******

<C>
Reprend l'exécution d'un programme en langage machine à partir de l'instruction située derrière le point d'arrêt qui l'a interrompue.

Le système aura auparavant rechargé les registres internes du 6809E avec leurs contenus respectifs avant l'interruption.
(Voir parag. <I>)

<D> D Zmem Déf: 168
Affiche sous forme hexadécimale et ASCII le contenu de la mémoire. Les octets qui ne pourront être transcrits sous forme ASCII seront symbolisés par des points. Notez que le bit 7 de parité est systématiquement mis à zéro pour cette transcription. Ainsi les octets $45 et $C5 seront traduits E en ASCII.

Exemple: *D C0A0—C0A7
=>
C0A0— 45 4E C4 46 4F D2 4E 45 ENDFORNE

<E> E Zmem, Data1/Data2 Déf: 256
Echange dans la zone mémoire définie par Zmem toutes les occurrences de Data1 par Data2. Ces deux chaînes de données doivent avoir la même longueur.

Exemples:
*E 1F40—1FFF, &768 / CB, %11010001
*E 2452—2551, "Trouve", Ø/ "Change", Ø

L'adresse de chaque occurrence trouvée est affichée avant le remplacement. De plus, les pauses et interruptions par <espace> et <STOP> sont valides.

<F> F Zmem, Data Déf: 256
Recherche et affiche les adresses de toutes les occurrences de Data dans la zone mémoire Zmem.

Lorsque le Moniteur vient d'afficher une adresse, il reste en attente d'une frappe de touche de votre part. Si vous désirez interrompre la recherche, pressez <STOP>, sinon frappez n'importe quelle autre touche. La fin de la recherche vous sera signalée par le message "Fin de recherche".

Exemple: *F C0A0—, "LOCAT", C5
=>
Occurrence en $C10D

<I> I 0 = adr..., 7 = adr
Installe les points d'arrêt numérotés de 0 à 7 aux adresses adr que vous lui spécifiez. Ces adresses ne doivent pas appartenir à la ROM du MO5 (adr<$C000). La rencontre de l'un de ces points d'arrêt dans un programme en langage machine lancé par la commande <X> provoquera l'interruption de son exécution et un retour au Moniteur. Celui-ci sauvegardera et affichera le contenu des registres internes du processeur. Vous pourrez reprendre l'exécution derrière le dernier point d'arrêt rencontré par la commande <C>.

Exemple: *I 0=25AA, 5=26CB
=>
I? est une fonction associée à la commande <I> qui permet de visualiser les adresses courantes des sept points d'arrêt possibles.

<J> J Adr

Effectue une exécution ligne à ligne à partir de Adr en affichant chaque instruction après son traitement ainsi que le contenu des registres internes. Le Moniteur forcera une pause après la première d'entre-elles. (Cf. 'PRESENTATION DU MODE DE COMMANDE', contrôle d'exécution).

Seule l'instruction SWI ne sera pas exécutée mais simplement affichée en raison de ses fonctions particulières dans la gestion interne du MO5.

Exemple: *J EA1A

```
        =>
EA1A=   A6 80     LDA  ,X+
S=9FFF  CC=04/Z  A=00  B=00
OP=21  X=0001  Y=0000  U=1F40  PC=EA1C
```

(Cf. <R> pour le détail de l'affichage des registres).

Nota: Avant toute exécution de la commande <J>, les registres internes du 6809E sont chargés avec le contenu des registres accessibles par <R=>.

<K> K FichO

Lit sur cassette le fichier binaire nommé FichO et installe son contenu en mémoire à son emplacement d'origine.

Exemple: *K "Graphe.BIN"

Nota: L'extension BIN est prise par défaut pour la recherche si aucune autre n'est précisée.

<L> L Zmem Déf: 36

Désassemble et liste le contenu de la mémoire en utilisant la syntaxe et les mnémoniques spécifiques au processeur 6809E.

L'affichage est décomposé en 4 champs successifs:

/Adresse du premier octet de la ligne.
/Octets désassemblés en hexadécimal.
/Mnémonique 6809E.
/Opérande de l'instruction.

Les valeurs exprimées en hexadécimal sont précédées par un "$".

Exemple: *L F000

```
        =>
F000=   7F 22 00  CLR  $2200
: : : :
F023=   37 20     PULU Y
```

Particularités:

Les instructions en Page Directe sont symbolisées par le caractère "/" en tête du champ d'opérande.

Les déplacements 5 bits en mode indexé sont signalés par leur expression décimale au contraire des offsets 8 et 16 bits affichés en hexadécimal donc précédés de "$".

En raison de la fonction spéciale des interruptions dans la gestion interne du MO5, l'instruction SWI requiert un paramètre donné sur un octet à la suite du code-opération. La nouvelle syntaxe de SWI sera ainsi: SWI # paramètre

Le mode immédiat est caractérisé par le symbole "#" placé en tête du champ d'opérande.

Le mode étendu est indiqué par un champ d'opérande constitué d'une simple adresse (ou étiquette. Cf. <A>).

Les opérandes indirects sont encadrés par des parenthèses.

<M> M Zmem, adr

Recopie la zone mémoire donnée à partir de l'emplacement adr. Une confirmation vous sera demandée par le Moniteur pour exécuter la commande. Entrez simplement (O)oui ou (N)on pour valider ou refuser l'exécution.

Exemple: *M FC9E–FF7D, 0

<O> O

Cette commande vous permet de retrouver les adresses de début et de fin d'ODIN, et l'emplacement de sa zone de stockage du Source.

Exemple: *O

```
      =>
ODIN 2.1    Odin:   5EAØ–927E
            Source: 25AA–5E9F
```

<P> P +/–

Tous les affichages opérés par les commandes Moniteur peuvent être envoyés sur imprimante en même temps qu'à l'écran. Entrez P+ pour connecter la sortie imprimante, P– pour la supprimer.

<Q> Q

Quitte le Moniteur et rend la main à BASIC. Si vous désirez revenir sous Moniteur, entrez "MON". Vous réintégrez ainsi le mode commande du moniteur SANS AUCUNE PERTE du texte Source éventuellement présent en mémoire.

<R> R registre=valeur,...

Permet de modifier le contenu des registres internes du processeur avant une exécution lancée par la commande <X>. Ils sont au nombre de 9: S, CC, A, B, DP, X, Y, U, PC. Tous, hormis PC et S, peuvent être modifiés par cette commande.
Tapez simplement le nom et la valeur du registre séparés par "=". Plusieurs attributions de valeur peuvent figurer sur la même commande, séparez-les simplement par une virgule.

R?

est une fonction associée à la commande <R>. Elle permet de visualiser le contenu de chaque registre.

Exemple: R B=EF, X=FDED, U=FBC1, CC=FF,?

```
      =>
S=ØØØØ CC=FF/EFHINZVC A=00 B=EF
DP=21 X=FDED Y=0000 U=FBC1 PC=0000
```

Comme vous avez pu le constater dans l'exemple précédent, un "?" introduit à la fin des paramètres d'attribution termine la commande en provoquant l'affichage du contenu des registres.
Cette facilité de contrôle est également disponible pour <I>.
De plus, le registre CC étant le registre d'état du 6809E, nous avons détaillé son contenu en affichant derrière un "/" les symboles correspondant aux drapeaux positionnés.

Nota: A l'initialisation, les registres sont remis à zéro excepté DP qui prend la valeur $21. Après une exécution de la commande <X>, le registre S passera à $9FFF.
Si vous utilisez la ROM MO5 dans vos programmes en langage machine, évitez de modifier le contenu de DP en raison des résultats pour le moins surprenants que cela pourrait engendrer.

<S> S Zmem Réf.: 256

Stocke en mémoire les données demandées par les entrées-ligne suivantes.

Exemple: *S 9F80–9F8D

=>

9F80–: 5A, 5B, 5C, "XYZEF" <ENTREE>
9F88–: &768 <ENTREE>
9F8A–: <ENTREE>

Lorsque vous aurez entré les données après l'appel du Moniteur représenté par l'affichage "adr–:", celles-ci seront stockées à partir de adr.
Si vous désirez stopper la commande avant terme, rendez une entrée-ligne vide.
Nota: Si l'espace mémoire défini par Zmem est insuffisant pour recueillir tous les octets présents sur l'entrée-ligne, le message "Stop: rang n" interviendra, où n est le rang sur la ligne du dernier octet qui aura pu être stocké.

<T> T Zmem, Data

Remplit la zone avec Data. Une confirmation vous sera demandée pour cette commande (Cf. <M>).

<U> U

Cette commande permet de conserver en mémoire une séquence d'ordres souvent appelée par l'Utilisateur. Toutes les commandes Moniteur peuvent être entrées pour mémorisation à la suite du caractère ">" envoyé par le système.

Exemple: *U

=>

>LF000–: 0: D2452–2551: ..: etc.

La chaîne ainsi obtenue ne devra pas excéder 128 caractères. Pour exécuter cette séquence entrez simplement XU (execute U).

U?

est une fonction associée à la commande U. Elle permet de visualiser le contenu courant de la séquence Utilisateur. Cette fonction est appelée d'office lors d'une commande XU.

<W> W FichO, Début, Fin, Exec

Permet de sauvegarder sur cassette le contenu de la zone comprise entre les adresses Début et Fin. La structure interne du fichier est identique à celle obtenue par la commande SAVEM de BASIC. Exec sert à donner l'adresse d'exécution pour une commande LORDM,,R ultérieure.

Exemple: *W "ASCII", FC9E, FF07, FC9E

Nota: L'extension ".BIN" est utilisée par défaut lorsqu'aucune autre n'a été donnée.

<X> X adr

Lance l'exécution d'un programme en langage machine à partir de adr. La reprise de contrôle du Moniteur se traduira par l'appel de la fonction <R?>.

Exemple: *X E8FC

Nota: Les contrôles de syntaxe derrière l'opérande adr ont été levés pour vous permettre de passer des paramètres ainsi récupérables par votre programme. Cette faculté existe aussi pour <J>.

<Y> Y 0...,7

Efface les points d'arrêt déclarés par la commande <I>. Il vous suffit de les identifier par leurs N^{os} (0–7).

<Z> Z Zmem, adr

Cette commande permet de reloger n'importe quel programme machine délimité par Zmem à partir de adr. Il s'agit d'une relocation standard qui ne modifie que les branchements absolus en fonction de la nouvelle adresse d'implantation, exception faite des sauts en Page Directe.
Si un code-op incorrect était rencontré durant la relocation, celle-ci serait arrêtée et le message "Err – CI" (Code-op illicite) apparaîtrait.

Z ODIN, adr

Cette commande vous donne la possibilité de reloger la zone programme du système Odin complet.
Seules les tables du système ne sont pas mobiles pour des raisons évidentes de fiabilité, mais elles ne représentent que 2,5 Ko en haut de la mémoire. A la fin de cette relocation, le Moniteur transféré prendra la main.
ATTENTION: Une relocation du système entraîne irrémédiablement une perte totale du texte source de l'Editeur. Le logiciel est d'ailleurs complètement réinitialisé (perte des points d'arrêt, de la séquence Utilisateur, etc...).

<!> !

Ce symbole représente le vecteur Utilisateur qu'il vous est possible de définir sous Moniteur de la manière suivante:

!=adr

Ultérieurement, l'utilisation du symbole "!" en tant que commande provoquera un saut à l'adresse adr.

!?

Est une fonction associée qui permet de visualiser l'adresse affectée au symbole "!".

Exemple: *!=21F9: !?: !
=> 21F9
Error 02

<#> #Encre, Fond

Cette commande vous permet de redéfinir les couleurs d'encre et de fond utilisées par le Moniteur exclusivement. Encre et Fond sont des valeurs entières allant de 0 à 15, données en décimal, qui correspondent aux codes des 16 couleurs disponibles sous BASIC:

0–Noir	8–Gris
1–Rouge	9–Rouge clair
2–Vert	10–Vert clair
3–Jaune	11–Sable
4–Bleu	12–Bleu clair
5–Magenta	13–Rose
6–Cyan	14–Bleu pâle
7–Blanc	15–Orange

Exemple: *#3,0
donne un fond noir à encre jaune.

<@> @ expression

Cette commande permet d'accéder au calculateur du module Assembleur. Il est conçu pour évaluer des expressions relativement complexes, dans les trois bases Hexa/Binaire /Déci disponibles sous Moniteur.
Le calculateur reconnaît 17 opérateurs au total, dont voici les symboles, les hiérarchies dans le calcul et les fonctions:

(Le chiffre entre parenthèses est d'autant plus élevé que la priorité de traitement de l'opérateur est grande. Si plusieurs d'entre-eux ont même priorité, l'évaluation se fera de gauche à droite).

```
!<      (5) Décalage gauche C<-xxxxxxxx<-0
!>      (5) Décalage droit   0->xxxxxxxx->C
!L      (5) Rotation gauche     -xxxxxxxx<-C
                               !            !
                               |----->------|
!R      (5) Rotation droite     C->xxxxxxxx-
                               !            !
                               |-----<------|
!R      (4) Et logique                And
!O      (3) Ou logique, inclusif      Or
!E      (3) Ou logique, exclusif      Eor
!C      (6) Complémentation à 1       Com
!N      (6) Complémentation à 2       Neg

!M      (5) Modulo
↑       (6) Exponentiation
=       (1) Egal à          ] Vrai rend  -1
?       (1) Différent de    ] Faux rend   0

+       (5) Addition
-       (2) Soustraction
*       (2) Multiplication
/       (5) Division
```

De plus, le calculateur reconnaît deux types d'opérandes supplémentaires venant s'ajouter aux valeurs numériques abordées dans la section "PRESENTATION GENERALE DU MODE DE COMMANDE". Le premier d'entre eux permet de représenter la valeur ASCII d'un caractère en le faisant précéder d'une apostrophe dans l'expression. Enfin, il vous sera possible d'entrer les étiquettes définies lors du dernier assemblage, que le calculateur remplacera par leur valeur associée pour évaluer l'expression.

Nota : En raison de la possibilité d'entrer ces étiquettes dans une expression, les valeurs hexadécimales devront être précédées de "S" pour lever tout ambiguïté quant à la nature de l'opérande traité. (SFACE = hexa. - FACE = symbole).
Les valeurs décimales peuvent au contraire être entrées sans préfixe "&" dans une expression de la commande <@>.

Exemple :

```
*@,$FBCE!< + 49991!M71
=>$FFA7
    63395
    11110111 10100011

*@,Label-1=Label+3-4
=>$FFFF
    65535
    11111111 11111111

*@,%1110*2[8+'Z'
=>$0E5A
    3674
    00001110 01011010
```

Comme vous avez pu le constater sur ces exemples, le résultat fourni par le calculateur est donné dans les trois bases du Moniteur (Hexa, Déci, Binaire). La commande <@> fait ainsi office de routine de conversion :

```
*@$EEDD
->$EEDD
   61149
   11101110 11011101
```

Enfin, celle-ci vous permet de connaître la valeur d'un symbole défini lors du dernier assemblage :

```
*@CHGET
->$21AC
   8620
   00100001 10101100
```

<//> /

Ce symbole est le commutateur qui vous permet de sélectionner l'un des deux modules de travail : Moniteur <-> Editeur/Assembleur. Utilisé sous Moniteur, il donnera le contrôle du mode de commande à l'Editeur. Nous vous rappelons qu'à cet instant, un autre jeu de commandes - concernant principalement la création et la gestion d'un texte Source - deviendra actif en remplacement du précédent décrit dans cette section.

LES COMMANDES DE L'EDITEUR

<A> A [S[Obj], [Bloc]]

Donne le contrôle au module Assembleur pour l'assemblage du texte présent en mémoire. Si Bloc est précisé, la constitution de la table des symboles et l'assemblage ne s'exécuteront que sur cette zone. En l'absence de délimitation, tout le texte en mémoire sera assemblé. Si Obj est donné, l'assembleur n'implantera pas le code-objet en mémoire durant la seconde passe, mais l'écrira sur cassette après confirmation de votre part, une fois terminé l'assemblage du texte.

De plus, il existe une variante de la commande d'assemblage appelée par le symbole-clé "A$". Elle indique à l'assembleur de constituer sa table de symboles à la suite de la table déjà présente en mémoire, et non pas en remplacement de celle-ci. Lorsque vous saurez qu'il est parfaitement possible de sauvegarder et de fusionner ces tables grâce aux commandes <W$> et <K$> d'ODIN, vous apprécierez la souplesse et la puissance de la programmation modulaire.

Exemples : A 150-350

```
A$ "objet"
A "objet", 100-1300
```

Nota : La commande <K$> fusionne une table écrite sur cassette avec une autre en mémoire, elle ne l'élimine donc pas. Si vous désirez supprimer la table résidente, utilisez la commande <V$>.

Pour plus de détails sur les fonctions disponibles et les écrans rencontrés sous Assembleurs, lisez la section 'DESCRIPTION DE L'ASSEMBLEUR'.

<C> C Bloc, lgn [,incr]

Cette commande copie le bloc de ligne spécifié à partir de lgn avec l'incrément incr. Le bloc de référence est conservé intact.

Exemple : &C 10—300,2000,5

<D> D Bloc

Détruit le Bloc concerné. Une confirmation vous sera demandée avant toute exécution de cette commande.

Exemple : &D 1000.50

<E> E [lgn][,[incr]]

Cette commande permet d'accéder au mode d'édition directe du texte source qui vous a été détaillé dans la section 'LE MODE D'EDITION'.

Quand vous donnez Lgn, la page d'édition commence par la ligne correspondante. Si la ligne n'existe pas, l'éditeur utilisera la première ligne au N° supérieur à celui que vous aurez entré. De plus, l'omission de lgn donne accès au début du texte, et lgn supérieur à tout N° existant représente sa fin.

Le paramètre ",inc" met l'éditeur en mode de numérotation automatique des lignes pour la saisie rapide d'un programme. Ainsi, le N° donné définit à la fois la ligne de début d'écran lorsqu'elle existe, et la ligne de base pour l'incrémentation.

Notez que la seule présence de la virgule dans la commande initialise ce mode ce qui vous permettra de faire appel à l'incrément par défaut.

Exemple : &E 500
&E 300,5
&E ,

<F> F [Bloc,]"chaîne"

Recherche et affiche toutes les lignes comportant la chaîne donnée. La comparaison se fera sur tous les champs des lignes incluses dans le Bloc à l'exception de celui des N°. De plus, la recherche d'une chaîne appartenant à deux champs adjacents est possible, en marquant la séparation par un espace blanc. Enfin, si Bloc est omis, la recherche concernera tout le texte en mémoire.

Exemple : &F 15000—,"ODIN"
&F 7.30,"ORG $5CCE"
&F "Loriciels"

<K> K FichO
K$FichO

Cherche à partir de la cassette le fichier source indiqué. Cette opération supprime l'ancien texte résident. Si une erreur E/S survenait pendant le chargement, la mémoire serait alors vide de texte.

Exemple : &K "SOURCE.ASM"

Nota : L'extension par défaut est ici ".ASM".

Cette commande inscrit FichO près de l'indicateur "Source :" désignant ainsi le nouveau fichier de travail.

Il existe une autre fonction attachée au symbole <K>. Elle est appelée par l'adjonction d'un "$" derrière la commande. Elle n'opère plus le chargement d'un fichier source mais permet de récupérer une table des symboles issue d'un autre assemblage préalablement sauvée sur cassette. Son but est de fusionner la nouvelle table à une autre déjà résidente en mémoire. Le nombre de chaînages que vous pouvez demander n'est limité que par la place mémoire encore disponible. Grâce à la commande <AS>, il vous sera possible d'exploiter n tables différentes plus celle constituée à partir du module Source résident.

Exemple : &KŞ "Symboles.SYM"
La commande <K> supprimant la table résidente, il est impératif de charger le Source avant les tables de symboles s'il y a lieu. De plus, si l'un des symboles chargée lors d'une commande <KŞ> est identique à l'un de ceux déjà résident en mémoire, le premier remplacera le second dans la table. Sachez de plus qu'en l'absence de Source, une commande <KŞ> implantera la table au début de la zone réservée au texte.
Nota : L'extension par défaut est ".SYM" pour une table de symboles.

<M> M FichO [,lgn[,incr]]
Fusionne le fichier indiqué avec le texte source résident. Il existe deux types de fusion opérés par cette commande : soit lgn est donné, et le fichier sur cassette sera installé à partir de ce N°, suivant l'incrément spécifié ; soit aucun paramètre de ligne n'est fourni, et les lignes du Fichier chargé reprendront leurs N°s d'origine.
Exemple : &M "Fusion",100,20
&M "Origine"
Nota : Si une ligne, quelque soit lé mode de fusion du fichier, avait un N° identique à une autre déjà résidente, elle viendrait remplacer cette dernière en mémoire texte.
Nota : L'extension par défaut est ".ASM".

<N> N lgn[,incr]
Renumérote tout le texte partant de lgn et suivant l'incrément incr.
Exemple : N 20,1

<O> O
Cette commande est identique en nom et en fonction à celle disponible sous Moniteur.

<P> P [Bloc]
Envoie sur imprimante la partie du source spécifiée. Si Bloc n'est pas donné, toute la mémoire texte sera listée.
Exemple : P 700—920

<Q> Q
Assure la même fonction que son synonyme du Moniteur. Si vous désirez revenir sous Editeur à partir de BASIC, entrez simplement "ASM".
Nota : Nous rappelons ici que la sortie Q n'altère pas le contenu de la mémoire texte de l'Editeur, ni les paramètres courants d'ODIN.

<R> R [Bloc,]"Trouver"/"Remplacer"
Remplace les occurrences de la chaîne de gauche par la chaîne d'échange à droite. En début d'exécution, ODIN vous demandera si vous désirez remplacer toutes les occurrences, ou seulement quelques-unes par le message "Total?(O/N)?"
Si vous répondez oui, le remplacement se fera automatiquement. Dans le cas contraire, une confirmation vous sera demandée pour chacun d'eux. Comme auparavant, répondez O pour que celui-ci soit opéré et N pour invalider et continuer la recherche. De plus, il vous est possible d'arrêter la commande en exécution par <STOP> aussi bien pendant une recherche que durant une validation. Dans ce dernier cas, le remplacement proposé est refusé d'office.

<S> S
Trie par ordre alphanumérique et affiche la table des symboles résidente en mémoire. Cette commande est un autre moyen pratique (Cf. <@>).

<T> T Bloc, lgn[,incr]

Transfère le bloc donné à partir de lgn. Cette commande correspond à la fonction <C>opie terminée par une suppression du bloc d'origine. Ainsi, elle ne peut s'effectuer que si la place mémoire restante accepte l'ajout d'un bloc égal à la zone de référence.

Exemple: &T 100—200,800,5

<V> V [[Lomem],[Himem]]
VS

Dans le premier cas (V) la mémoire texte est réinitialisée après confirmation de votre part. Si vous spécifiez les paramètres entre crochets, il vous sera possible de délimiter la zone mémoire attribuée au texte source. Lomem représente l'adresse la plus basse de la mémoire disponible pour le stockage, Himem la plus haute. Ces adresses ne pourront chevaucher les zones programme et données d'ODIN, ni descendre sous l'adresse £25AA critique pour la gestion interne du MO5. De plus, Lomem devra être inférieur ou égal à Himem.

Nota: Il vous est possible d'omettre l'un des deux paramètres. L'ancien sera alors pris par défaut.

Exemples: &V 4F00,5F00
&V ,5F4F

Cette commande conservera la table des symboles résidente sauf si l'adresse Lomem est supérieure à l'adresse de début de la table.
Enfin, la commande VS permet de supprimer la table des symboles résidente. Elle nécessitera aussi confirmation de votre part.

<W> W [FichF][,Bloc]
WS FichO

Ecrit sur cassette le Bloc spécifié sous le descripteur FichF. Si Bloc n'est pas présent, tout le texte sera sauvegardé. Si FichF est omis, ODIN prendra le nom de fichier inscrit après l'indicateur "Source:".

Exemple: &W "Source",900—
&W

Après cette commande, FichF viendra s'inscrire près de l'indicateur "Source:" comme identificateur du fichier de travail.
La commande dérivée WS sauvegardera la table des symboles résidente. Elle constitue le complément des fonctions <K$> et <A$> concernant ce type de fichier.

Nota: L'extension par défaut est "ASM" lors d'une commande <W> et "SYM" lors d'une commande <WS>.

<X> X Zmem, lgn[,incr]

Cette fonction permet de transformer du code-objet de la mémoire en texte source compatible avec l'Editeur-Assembleur d'ODIN. Cette commande récupèrera même les symboles appartenant à une table résidente, si vous autorisez l'échange adresses-symboles par la commande <A+> du Moniteur.
Une seule limitation: le code rencontré doit correspondre à l'une des instructions valides du 6809E.
La commande <X> permet entre autre de récupérer les programmes objets que vous avez pu constituer auparavant sans l'aide d'un assembleur, et ainsi de profiter de la facilité de manipulation qu'apporte un Editeur. De plus, si vous pensez à sauvegarder régulièrement vos tables de symboles par la commande <WS>, vous pourrez toujours en cas de destruction du Source reconstituer un nouveau texte à partir d'un fichier objet et d'une table chargée par la commande <K$>.

Exemple: &X F000—F0B2,500,1

Nota : Chaque ligne entrée sera listée à l'écran. De plus, si son N° est identique à celui d'une ligne déjà présente en mémoire, elle viendra l'y remplacer.

</> /
Cette commande permet de passer du module Editeur sur lequel vous travaillez au module Moniteur d'ODIN. Vous reconnaîtrez le changement de contrôle au prompt "*" du Moniteur qui remplacera le "&" de l'Editeur.

<#> # encre cmd, fond cmd, fond edt
Cette commande est similaire à son synonyme du Moniteur. Notez cependant l'intervention du troisième paramètre qui ne définit plus une couleur de fond du mode de commande, mais la couleur de la fenêtre du mode d'édition directe du texte.

Exemple : &# 14,0,4

Nota : Les codes correspondants aux couleurs sont identiques à ceux répertoriés dans la description de la commande <#> du Moniteur.

<.> .lgn
Nous avons ajouté cette commande pour vous permettre de retrouver aisément les adresses d'implantation en mémoire texte de n'importe quelle ligne. Si cette commande donne effectivement la possibilité de manipuler directement le texte à partir du Moniteur, nous vous conseillons fortement de ne pas modifier vous-même la structure de vos lignes de mémoire. En effet, un codage incorrect est susceptible de provoquer la déroute de l'Editeur ainsi que celle de l'Assembleur, et il en résulterait vraisemblablement une perte du Source résident.
Cependant, cette commande est fort utile pour connaître les adresses des première et dernière lignes du texte.
Ainsi : .0 donne l'adresse la plus basse
.65535 donne la plus haute.

<@> @ expression
Ceci est la commande écho sous Editeur de la commande <@> sous Moniteur. Tous les détails sur cette commande vous sont donnés dans la section précédente 'LES COMMANDES DU MONITEUR'.

DESCRIPTION DE L'ASSEMBLEUR

Introduction

Le module assembleur d'ODIN est conçu pour générer le code machine valide sous 6809E correspondant au texte source que vous aurez pu créer avec l'Editeur.
Co-résident avec les modules Moniteur et Editeur, il autorise les assemblages directement en mémoire, permettant ainsi un test et une mise au point immédiats du programme objet réalisé. Si ce programme ne peut cohabiter en mémoire centrale avec ODIN, le module assembleur vous permettra tout aussi bien de choisir l'assemblage sur cassette, le code-objet étant alors écrit dans un fichier exécutable par la commande LOADM " ",,R de BASIC.

La commande d'assemblage

Comme nous avons pu le voir dans la section descriptive de l'Editeur, un assemblage se déclenche par la commande:

A [S[Obj],[Bloc]]

Nous ne reviendrons pas sur la signification des paramètres de la commande <A> qui est détaillée dans la section 'LES COMMANDES DE L'EDITEUR'.

Les phases de l'assemblage

L'assembleur d'ODIN procède au minimum à deux passes distinctes lorsqu'il traduit un texte Source en code Objet. La première regroupe toutes les étiquettes définies par l'utilisateur dans une Table des Symboles, calculant pour chacune d'elles une valeur de référence. Cette Table est toujours implantée à la suite du texte Source en mémoire. Une fois cette phase terminée, la seconde passe commence, générant le code Objet proprement dit. Elle utilise les informations de la Table des Symboles pour évaluer les expressions symboliques du Source.

Enfin, si vous avez demandé une sortie de l'Objet vers la cassette, une troisième passe est activée pour sauvegarder le code machine généré en passe 2 dans le fichier choisi lors de la commande.

Sachez de plus qu'à tout moment vous pouvez constater la progression de l'assemblage par deux indicateurs spéciaux:

'Passe n' désigne la phase courante de l'assemblage parmi les trois possibles qui sont décrites ci-dessus.

'nnnnn/total' représente l'état de progression de la phase courante. Cet indicateur affiche modulo 50 le nombre de lignes déjà traduites en code machine sur le nombre total de lignes à traiter. Notez que ce total ne tient pas compte de la possibilité de réduire de façon interne le nombre de lignes à assembler par l'insertion de la directive 'END' dans le texte Source.

Structure d'un fichier Source

Un fichier ou programme Source est constitué d'un ensemble de lignes créées grâce au module Editeur d'ODIN. Ces lignes sont composées de 4 champs distincts qui sont respectivement:

Le champs de N° de ligne, celui-ci est ignoré lors de l'assemblage pour la génération du code Objet.

Le champs d'Etiquette, contient un symbole généralement optionnel. Celui-ci est une chaîne d'au plus 6 caractères qui commence obligatoirement par une lettre. De plus seuls les lettres et les chiffres sont valides dans une étiquette symbolique.

Cependant, majuscules et minuscules sont différenciées par l'assembleur. Ainsi, "ODIN" et "odin" sont deux étiquettes distinctes. Enfin, les espaces blancs sont supprimés des symboles lors de la validation des lignes en mode d'édition.

Une étiquette doit être unique dans le programme assemblé, chaque caractère étant testé pour vérifier la différence entre deux symboles. Si l'assembleur rencontre plusieurs fois le même symbole au cours du traitement, le message "Symbole redéfini" est affiché. Pourtant, il vous est possible de redéfinir la valeur attribuée à un symbole à condition de faire appel à la directive 'SET' en lieu et place de 'EQU'. De plus, une étiquette définie dans un bloc conditionnel non assemblé ne participe pas à la Table des Symboles, et ne provoque donc pas d'erreur de redéfinition si cette même étiquette existe déjà dans la Table.

La valeur assignée à chaque étiquette est celle du pointeur programme calculé par l'assembleur, sauf lorsque le symbole est utilisé avec les directives 'EQU' ou 'SET'.

Si une référence à l'un des symboles déclarés est faite, sa valeur assignée lui est substituée. Cependant, une référence à une étiquette non définie engendre un message "Symbole non déclaré" durant la passe 2. De plus, il existe une autre condition d'erreur signalée par

l'assembleur lors de la passe 1 quant à l'utilisation des étiquettes symboliques. Elle est provoquée par une référence à un symbole effectivement défini, mais dont la valeur n'a pu encore être calculée alors qu'elle doit influencer le pointeur programme lui-même. Ainsi, le programme :

```
        ORG     DEBUT
        —
        —       instructions
DEBUT   EQU     $6ØØØ
```

engendrera le message "Référence impossible".

Le champ du mnémonique, contient un code opération requis pour toute ligne à assembler, à l'exception de lignes de commentaires. Ce mnémonique est soit l'un de ceux reconnus par Motorola pour le processeur 6809E, soit l'une des directives d'assemblage spécifiques à ODIN qui vous seront décrites plus loin.

Le champ d'opérande, contient une expression parfois nécessaire aux instructions du 6809E ou aux directives d'assemblage. L'assembleur y reconnaît la syntaxe propre à chaque instruction du 6809E définie par le constructeur du micro-processeur, ainsi que les paramètres requis pour chaque directive. Notez qu'une tentative de référencer dans le champ d'opérande une étiquette symbolique non définie dans le programme provoquera une erreur "Symbole Non Declare" lors de la passe 2.

Particularités syntaxiques

Un certain nombre de caractères spéciaux introduits en tête du champ d'opérande indique à l'assembleur le mode de codage souhaité pour les instructions licites du 6809E.

"/" marque l'utilisation en page directe. L'assembleur ne retient que l'octet de poids faible du résultat de l'expression. ($2E5C <=> /$5C)

"<" force le format de l'opérande à 8 bits. Ne s'utilise qu'en mode indexé.

">" force ce même format à 16 bits.

Nota : L'assembleur tentera toujours d'utiliser le format le plus court si aucun n'est spécifié (5 ou 8 bits, 16 si nécessaire).

"#" indique le mode immédiat.

Si aucune entête n'est utilisée, et si le mode n'est pas indexé, l'instruction sera codée en adressage étendu sur deux octets d'opérande.

Les opérandes indirects devront être encadrés par des parenthèses.

— Il existe également des modifications du champ d'opérande ouvrant de nouvelles possibilités ou simplement rendues nécessaires par votre MO5.

Ainsi, toutes les instructions d'empilement et de dépilement, de transfert et d'échange, peuvent être paramétrées par une valeur numérique représentée dans un mode immédiat :

```
        TFR     #$23
        PULS    #$FF
```

sont alors valides.

Enfin, la dernière particularité réside dans la syntaxe adoptée par l'Assembleur pour l'instruction SWI. En effet, celle-ci doit être suivie d'une opérande immédiate sur un octet car il est impossible d'utiliser cette fonction d'interruption sur MO5 sans spécifier ce paramètre.

Calcul d'expressions numériques

L'assembleur accepte de calculer des expressions complexes introduites dans les champs d'opérandes. Leur résultat peut ainsi représenter une adresse ou tout autre paramètre numérique d'une instruction ou d'une directive d'assemblage.

La syntaxe de ces expressions et les opérateurs disponibles sont identiques à ceux valides lors d'une commande <G> décrite dans le Moniteur.

Un non-respect des caractéristiques qui y sont indiquées engendre le message "Expression incorrecte" durant la passe 2.

Le contrôle d'assemblage

Des facilités de contrôle du processus d'assemblage ont été incluses pour vous dans cet assembleur. Elles sont de deux ordres :

- Pause et interruption d'assemblage

Une vérification continue du clavier est opérée par votre MO5 lors des 2 (ou 3) passes d'un assemblage. Si une pression sur <STOP> est détectée lors de l'assemblage et quelque que soit son stade, un retour inconditionnel est opéré vers l'Editeur. De plus, il vous est possible d'utiliser la barre d'espacement pour générer un assemblage instruction par instruction, une pause intervenant alors après son traitement avant de poursuivre avec la suivante. Pour continuer ce mode, frappez de nouveau sur la barre d'espacement ; pour reprendre le processus normal, entrez n'importe quelle autre touche sauf <STOP>.
Il existe un autre moyen de générer une pause inconditionnelle lors de l'assemblage. Il suffit d'installer la directive WAIT à l'intérieur du Source. Sa rencontre provoque l'émission d'un beep sonore, et met l'assembleur dans une attente de frappe de touche de votre part pour reprendre son travail.

- Contrôle de l'affichage

Deux commandes permettent d'autoriser ou d'inhiber l'affichage du texte Source et de sa traduction en codes d'instructions hexadécimaux qui est effectué durant la passe 2 d'un assemblage.
La première commande est une directive que vous pourrez insérer dans le programme Source. Elle affecte l'affichage des lignes qui la suivent dans le texte. Sa syntaxe est :

LIST expression

Si le résultat global de l'expression est non nul, les lignes suivantes seront affichées avec leur traduction en codes hexadécimaux. Dans le cas contraire, aucun affichage ne sera plus opéré durant la passe 2, tout au moins jusqu'à la rencontre d'une autre directive LIST.
Il existe une seconde commande, celle-ci directement accessible au clavier durant la passe 2, qui vous permet d'inverser la condition courante de votre affichage. Par exemple, si vous avez inhibé tous les affichages par une commande LIST 0 intégrée au Source, il vous est possible de revenir au mode normal d'affichage en entrant :

CNT—L

L'opération inverse qui consisterait à empêcher les affichages suivants est bien entendu possible avec la même commande.

LES DIRECTIVES D'ASSEMBLAGE

Ces directives provoquent, lorsqu'elles sont rencontrées par l'assembleur, une opération visant au contrôle du processus d'assemblage lui-même, ou à la définition de données diverses ou d'étiquettes symboliques référencées dans le programme. Elles sont en outre représentées par des mnémoniques précis qui sont insérés comme les code-opérations habituels dans le texte Source.

DIRECTIVES INTERNES

ORG org expression

Cette directive ou Pseudo-opérateur sert à fixer une adresse d'origine pour l'implantation du code-objet généré durant un assemblage. L'expression qui est évaluée fournira donc l'adresse où sera stocké le premier octet généré. Si aucune directive ORG n'est donnée avant la génération d'un code-objet, l'adresse d'implantation par défaut sera celle de la fin de la Table des Symboles arrondie à la page mémoire supérieur (1 page = 256 octets). Ainsi, pour une Table des Symboles finissant en $52C0, cette adresse par défaut sera $5300.
Si vous utilisez une étiquette dans l'expression, assurez-vous que la valeur qu'elle représente a déjà pu être déterminée par l'assembleur au moment où vous y faites appel. Il est prudent à cet égard de toujours définir les étiquettes dont vous connaissez l'adresse par avance au début de votre programme Source (voir EQU & SET).
Vous pouvez définir plusieurs blocs d'instructions ou de données, implantés à partir d'adresses discontinues, grâce à la directive ORG. Ainsi, à chaque rencontre de ce Pseudo-op, l'assembleur initialisera la nouvelle adresse d'implantation des octets suivants avec le résultat de l'expression fournie.
Nota: Les effets de la directive ORG sont strictement identiques que vous ayez choisi l'assemblage avec Objet co-résident ou écrit sur la K7.

OBJ obj expression

Si le résultat global de l'expression est non nul, l'assembleur implantera en mémoire le code-objet généré. Dans le cas contraire, aucun stockage de l'Objet ne sera effectué. Cependant, si vous êtes en mode d'assemblage K7, le résultat des directives OBJ rencontrées est forcé à 1. Ceci ne signifie pas pour autant que le code sera co-résident avec ODIN et votre Source puisque sa sortie sera détournée de la mémoire vers la K7.
Notez qu'un refus d'implanter le code-objet généré par la directive OBJ 0 n'affecte en rien l'affichage de celui-ci lors de la passe 2.

EQU Etiquette equ expression

Cette directive assigne une valeur à une Etiquette qui pourra donc être référencée symboliquement au cours du programme qui l'a défini. Ce symbole doit être unique dans tout le texte Source; il reçoit la valeur de l'expression calculée par l'assembleur en passe 1.
La possibilité de remplacer les adresses absolues ou n'importe quelle donnée sur 1 ou 2 octets par un nom symbolique facilite grandement la mise au point et la modification ultérieure d'un texte Source. Ainsi, "ENVOI" est-il plus parlant que "$CE32" après six mois de rupture avec un logiciel!
De plus, il est plus pratique de changer une seule valeur assignée au symbole "Etoile" en tête d'un programme que de remplacer 20 à 30 "$2A" dans un texte de 15 ko... même si ODIN vous permet de faire des recherches et des remplacements globaux sur le dit texte...

SET Etiquette set expression

Attribue une valeur temporaire au symbole spécifié. A la différence de la directive EQU, cette valeur peut être remplacée de façon illimitée dans la suite du programme par de nouvelles assignations SET.
Nota: Si vous utilisez EQU pour attribuer une valeur à un symbole jusqu'alors assigné par SET, cette valeur devient fixe et toute tentative de la changer engendrera un message de redéfinition.

END end [expression]

Cette directive force l'arrêt de l'assemblage à la ligne courante lorsqu'elle est rencontrée en passe 2. La table des symboles est pourtant constituée avec les étiquettes déclarées dans l'ensemble du texte résident, ce qui permet de tester une zone du Source à l'assemblage dont les instructions référencent un bloc distinct non assemblé, et ceci, sans générer d'erreur d'indéfinition de symboles.

L'expression facultative qui peut suivre END précise l'adresse d'exécution du fichier Objet qui sera constitué lors de la passe 3 d'un assemblage avec sortie cassette. Si le mode choisi est celui de la co-résidence de l'Objet avec ODIN, cette adresse sera le nouveau vecteur utilisateur de la commande <!> du Moniteur.

DIRECTIVES D'ASSEMBLAGE CONDITIONNEL

Ce groupe de pseudo-opérateurs autorise la génération d'un objet en fonction de l'assemblage de blocs de texte Source alternatifs.
L'utilisation la plus courante de cette facilité consiste à établir un programme Source compatible avec des configurations différentes d'une même machine. Ainsi, la partie du Source correspondant à une configuration précise sera sélectionnée lors de l'assemblage au détriment des autres par un simple test conditionnel.
Une brève modification d'un paramètre défini au début du texte à assembler permet alors de choisir la configuration souhaitée. Ces opérations conditionnelles sont accessibles par le triptyque DO, ELSE et FIN.
Exemple:

```
00100    DISQUE EQU 1
00110    ; config. avec disque=1, K7=0
.....    ..........................................................................
10800        DO DISQUE
10810    ; suite d'instructions
10820    ; initialisant par exemple des
10830    ; paramètres d'accès au disque.
10840        ELSE
10850    ; instructions symétriques
10860    ; initialisant au contraire
10870    ; une configuration cassette.
10880        FIN
10890    ; sortie du bloc conditionnel.
```

DO do expression

Cette directive permet à l'utilisateur d'ouvrir un bloc conditionnel. Si le résultat global de l'expression est non nul, l'assembleur traitera les instructions qui suivent DO jusqu'au premier ELSE ou FIN. Si ce résultat est nul, les instructions suivantes seront ignorées jusqu'à ces mêmes limites.
Lorsqu'un DO intervient alors qu'un précédent est toujours en action, l'assembleur définit un sous-bloc conditionnel et sauvegarde l'état (nul/non.nul) du premier rencontré. Il sera conservé dans une pile des conditions (LIFO), et récupérable lorsque le sous-bloc courant aura été fermé par une directive FIN. Cette pile a 80 niveaux de profondeur, ce qui vous permettra d'ouvrir jusqu'à 80 blocs conditionnels imbriqués.
L'expression donnée est évaluée durant la passe 1, ce qui suppose qu'aucune référence à un symbole sans valeur encore assignée n'y soit faite.

ELSE else

Ce pseudo-opérateur permet d'inverser la condition courante d'un bloc déjà ouvert par une directive DO précédente. Elle offre ainsi une structure conditionnelle simple pour définir une alternative d'assemblage. Cette directive n'a aucun effet sur les états des conditions inférieures stockés dans la pile.

FIN fin

La directive de fin d'alternative clôt le bloc conditionnel courant ouvert par le précédent DO. Si une condition inférieure existe, son état est extrait de la pile et prend immédiatement effet. Sinon, l'assemblage retourne au mode inconditionnel.

Nota : Lors de l'affichage d'une ligne contenant les directives DO/ELSE, le résultat du test conditionnel est donné sous le champ d'opérande :

```
00220        DO  4/2=2
                 vrai
00221        ELSE
                 faux
```

Ici, la première assertion "4/2=2" est vraie, −1 est alors renvoyé ; l'état est donc "vrai". ELSE inverse cet état, qui prend ainsi la valeur "faux".

DEFINITIONS DE DONNÉES

Les directives suivantes permettent de définir des chaînes d'octets explicites ou des zones mémoire.

FCB fcb expression[,exp...]
Cette directive demande à l'assembleur de considérer les expressions fournies comme autant de valeurs entières codées sur un octet chacunes. Les octets ainsi obtenus sont mis à l'adresse du pointeur programme courant. Si une expression donnée est trop importante pour être codée sur un seul octet, une erreur "Opérande trop Grand" sera générée en passe 2.

FDB fdb expression[,exp...]
Cette directive effectue la même opération que le pseudo-opérateur FCB exception faite de l'ordre de grandeur des expressions qui est porté à deux octects.
Nota : Seule la virgule est autorisée pour séparer plusieurs expressions.

FCC fcc 'chaîne'
Cette directive considère la chaîne donnée comme une suite de codes ASCII correspondant à chaque caractère. Tous les octets ont leur bit de parité (bit 7) à 0.

FCI fci 'chaîne'
La fonction de FCI est jumelle de celle de la directive FCC exception faite de l'état du bit de parité du dernier caractère de la chaîne qui est mis à 1.

RMB rmb expression
Cette directive crée une zone mémoire réservée, à partir du pointeur programme courant, dont la longueur est déterminée en nombre d'octets par l'expression. Celle-ci doit être inférieure à 256 octets ; de plus, aucune initialisation à une valeur particulière ne sera faite sur cette zone.

DIRECTIVES EXTERNES

COPY copy expression
Sélectionne l'imprimante comme périphérique de sortie additionnel à l'écran si le résultat global de l'expression est non nul. Sinon, tous les affichages suivant de l'assemblage seront exclusivement dirigés vers l'écran.

TITLE title ['chaîne']
Si l'imprimante est sélectionnée, l'assembleur enverra la chaîne spécifiée toutes les 66 lignes imprimées comme titre courant. Cette directive remplace par sa chaîne l'éventuel titre déclaré par un TITLE précédent. Ainsi, vous pouvez supprimer un titrage en omettant la chaîne de paramètre dans la commande.

SKIP skip expression
Envoie le nombre de sauts de ligne avec retour en marge gauche spécifié par l'expression. Ce nombre doit être inférieur à 256.

PAGE page

Génère un saut de page, quelque soit le périphérique de sortie sélectionné. Celle directive est identique à la commande SKIP 72.

STR str expression

Cette directive est utilisée afin de créer une chaîne d'un même caractère qui sera envoyée vers le périphérique de sortie. Sa longueur est donnée par le résultat de l'évaluation de l'expression, toujours inférieur à 256 unités. Le caractère répété peut être choisi par la directive CHR. Par défaut, il est initialisé à l'étoile ('*').

CHR chr expression

Change le caractère de répétition de la directive STR par le symbole alphanumérique représenté par son code ASCII dans l'expression. Le calculateur acceptant des opérandes ASCII directes, vous pouvez définir la commande ainsi: CHR '&'.

Nota: Toutes les apostrophes que nous utilisons pour encadrer des chaînes dans les champs d'opérande peuvent être remplacées par des guillemets.

LA TABLE DES SYMBOLES

Lorsqu'il a terminé les deux premières passes, l'assembleur effectue le tri alphabétique et l'affichage optionnel de l'ensemble des symboles définis dans le texte Source. Il est possible que seule cette table des symboles vous intéresse; en ce cas insérez la directive LIST 0 en tête du Source et LIST 1 à sa fin.

Si vous désirez au contraire inhiber l'affichage de la table, entrez LIST 0 en fin de texte. Nous vous rappelons qu'il est également possible de faire directement à partir du clavier ces sélections, et ceci durant l'assemblage, en utilisant CNT—L.

Lorsque l'affichage est autorisé, il se fait sur 40 colonnes avec deux symboles listés par ligne, quelque soit le périphérique de sortie en service. Le nom symbolique est d'abord donné, puis son adresse hexa. Vous pourrez remarquer qu'un caractère précède parfois ce nom symbolique:

'*' indique que le symbole a provoqué une erreur de redéfinition pendant l'assemblage.

'?' révèle que l'étiquette a été définie mais jamais référencée, ce qui la rend pratiquement inutile.

Vous constaterez également que les étiquettes ayant généré une erreur du type "Symbole non déclaré" ne sont pas listées avec les autres. En effet, elles sont affichées après, avec l'en-tête "Sans Définition:" pour éviter une recherche fastidieuse dans une table des symboles parfois longue.

Ces étiquettes ne sont pas triées, elles apparaissent dans l'ordre où l'assembleur les a rencontrées. Même si plusieurs références sont faites à un symbole indéfini, celui-ci ne sera pourtant affiché qu'une fois. De plus, ce listage particulier est toujours effectué, y compris lorsque l'affichage a été inhibé car il rend compte d'erreurs.

Exemple d'affichages des symboles:

```
00006 Symboles définis:
AFCD     4ECD      *MULTI    398E
?NONREF  3B4F       PAGE     5E70
SYLOD    7140       TKOP     6FEC

Sans définition:
INCNU               SSDEF
OUBLI
```

Cette table résidera en mémoire jusqu'à sa suppression lors du chargement d'un nouveau Source, l'accroissement du texte résident, ou l'utilisation de la commande <V$> de l'Editeur.

LE RECAPITULATIF D'ERREURS

Comme son nom l'indique, ce récapitulatif n'a lieu que si l'assembleur a décelé des erreurs lors du traitement. (Dans ce cas, il est l'ultime opération effectuée lors d'un assemblage). Leur nombre est d'abord affiché puis une pause est générée. Une frappe de touche provoquera alors le listage des lignes incriminées et des types des erreurs rencontrées. Les lignes erronées sont listées par ordre croissant des N°.

```
Exemple de récapitulatif d'erreurs:
00002 erreurs
00700 symb      FCB     300,132,$40,"
Operande Trop Grand
00920           NEG     #$EC
Mode incorrect
```

La capacité de récapitulation a volontairement été limitée à 85 erreurs pour éviter d'immobiliser une place mémoire prohibitive. Celle-ci est largement suffisante à une utilisation normale. De plus, un nombre supérieur est généralement causé par un même type d'erreur répercuté sur le texte entier.

Par exemple, une directive ORG mal paramétrée pour un assemblage en mémoire provoquera autant d'erreurs "Adresse incorrecte" qu'il y a de lignes dans le Source!

ANNEXES

ANNEXE A

CARTE MEMOIRE

A titre indicatif, voici le découpage mémoire courant lorsque le système ODIN est chargé.

<Adresse>	<Utilisation>
0000	... Ecran, variables et pointeurs de gestion interne du MO5
2000	...
2800	Mémoire texte pour le fichier Source à l'initialisation
3600	Début de la zone programme d'ODIN à l'initialisation [Ces deux zones sont susceptibles de changer d'adresses après une relocation Système d'ODIN]
9280	Tables du système
9300	Zone de travail
9400	Pile système
9F00	Buffers divers
A000	... E/S et ROM du MO5
FFFF	

ANNEXE B

LES ERREURS DU MODE COMMANDE

En mode commande d'ODIN, une erreur est signalée par l'en-tête "Err –", suivi d'un code de deux lettres.

SN – Syntaxe générale incorrect. Une commande ne peut être reconnue, ou des paramètres superflus ont été introduits.

DB – Débordement de capacité de calcul. L'expression donnée ne peut être codée sur 16 bits.

DZ – Division par zéro rencontrée en mode calcul.

CI — Code-Opération Incorrect. Arrive lors d'une commande <Z> Moniteur ou <X> Editeur. Le système a rencontré un code sans correspondance avec une instruction valide du 6809E.

OM — Opérande Manquant. Un opérande servant de paramètre dans une commande a été omis.

OI — Opérande Incorrect. Les types de paramètres fournis ne correspondent pas à ceux demandés.

AI — Adresse Incorrecte. Provoqué par une tentative de donner une adresse d'implantation de programme ou de données mettant en péril l'intégrité de la zone programme ou donnée d'ODIN. (Cf. Annexe C - Adresse Incorrecte).

CL — Chaîne de caractères trop longue donnée dans un paramètre. La chaîne doit être inférieure à 128 caractères.

BV — Buffer de texte Vide. La commande de l'Editeur n'a pu agir sur une mémoire vide de texte.

LI — Ligne Inexistante. Un N° de ligne donné dans l'Editeur ne correspond à aucune ligne connue.

MP — Mémoire Pleine. Une commande de l'Editeur a tenté d'occuper plus de mémoire texte que de disponible.

EX — Expression Incorrecte. Survient lors d'un calcul <e> sur une expression mal formulée. En général, l'un des opérateurs n'a pu être identifié.

SY — Un Symbole non défini dans la table résidente a été référencé lors d'un calcul.

VM — Valeur Manquante. Une omission d'opérande avant ou après un opérateur a été constatée dans un calcul.

CX — Expression trop Complexe. La taille de l'expression donnée pour un calcul dépasse la capacité de la pile.

IL — Intervalle insuffisant entre deux lignes existantes pour en insérer de nouvelles. Survient lors d'une commande de copie ou de déplacement de bloc lorsque l'opération risque d'écraser des lignes présentes.

Enfin, une commande faisant appel à un périphérique (lecteur de K7, imprimante) peut provoquer une erreur qui sera alors représentée par un code décimal identique à ceux utilisés par BASIC. Vous pouvez le cas échéant vous reporter au manuel du MO5 pour en connaître la signification.

ANNEXE C

MESSAGES D'ERREURS DE L'ASSEMBLEUR

Contrairement au mode commande, les erreurs durant un assemblage sont données en clair pour faciliter la lecture sur un affichage mobile. Voici leur signification :

Expression incorrecte
Ce message est équivalent au code "EX" d'une erreur en mode commande. Il survient lorsqu'un calcul sur un champ d'opérande n'a pu être mené à bien à cause d'une erreur de syntaxe, une omission d'opérateur ou d'opérande. De plus, le débordement de capacité de calcul (DB) est lui aussi inclus dans cette signification.

Symbole Non déclaré
Cette erreur est signalée derrière une ligne dont le champ d'opérande contient une référence à un symbole qui n'a pas été défini dans le Source.

Opérande Trop Grand
Ce message est envoyé lorsque le résultat de l'évaluation est numériquement trop grand pour tenir sur le format demandé par l'instruction.

Mémoire Saturée
Ce message n'intervient qu'en passe 1, lorsque l'assembleur n'a plus assez de place en mémoire pour constituer le reste de sa table de Symboles. C'est ce que l'on peut appeler une erreur fatale, c'est-à-dire qu'elle annule l'assemblage et retourne à l'Editeur. Pour pallier à ce genre d'erreur très désagréable, il est possible de diviser votre texte en plusieurs modules plus petits que vous pourrez assembler séparément en réutilisant les tables des Symboles dont vous aurez besoin. Pour plus de détail sur cette facilité, reportez-vous à la section 'LES COMMANDES DE L'EDITEUR', paragraphes <W$>, <K$> et <A$>.

Référence impossible
Un assembleur travaille de manière séquentielle. Ainsi, il ne connaît jamais lors de sa première lecture du texte les éventuelles définitions qui peuvent être faites plus avant dans un Source. De plus, il doit déjà pouvoir évaluer sa ligne courante et donc modifier son pointeur programme en conséquence avant de savoir calculer l'adresse d'implantation de la prochaine instruction. Ce type d'erreur est précisément une tentative de référencer une étiquette symbolique non encore évaluée, et qui ne pourra l'être qu'après que ce champ d'opérande soit lui-même calculé! A ce choix cornélien, aucun assembleur ne pourra trouver une solution. Pourtant, nous avons tenté de faciliter votre mise au point en différenciant ce cas de l'indéfinition pure et simple par le message de référence impossible qui signifie que le symbole est effectivement déclaré ultérieurement dans le texte,mais que sa valeur n'a pu encore être calculée.
Ainsi, cette erreur survient également lorsque le symbole référencé appartient à un bloc conditionnel DO/FIN non assemblé.

Symbole Redéfini
Cette erreur intervient lorsque plusieurs définitions d'un même symbole ont été tentées tout au long du Source. Nous vous rappelons que la directive SET permet d'assigner des valeurs temporaires à des étiquettes, à l'exclusion de toute autre méthode.

Adresse incorrecte
Ce message indique que lors d'un assemblage avec implantation directe en mémoire de l'Objet, une tentative a été faite d'installer le code généré dans une zone protégée du système ODIN. Ces zones sont au nombre de 5: avant l'adresse $25AA - sur la zone programme d'ODIN - sur la mémoire texte qui contient le Source - sur la table des symboles courante - au-dessus de l'adresse $9280.
Cette erreur est strictement équivalente au code "RI" du mode commande.

Opérande Manquant
Ce message est engendré par une tentative d'omettre un opérande nécesaire à une instruction.

Condition non déclarée
Cette erreur survient lorsque l'assembleur rencontre une directive ELSE ou FIN alors qu'aucun bloc conditionnel n'a été ouvert par DO.

Opérande incorrect
Cette erreur indique qu'une donnée ou un caractère-clé ont été omis dans le champ d'opérande. Souvent, il s'agit d'une parenthèse d'indirection non refermée, abusivement ouverte ou vice-versa...

Pile saturée
Ce type d'erreur est tout à fait rarissime car il traduit un dépassement de la capacité de stockage des états conditionnels lorsque vous ouvrez sans jamais les refermer des blocs successifs par la directive DO. Vous disposez pour cela de 80 niveaux d'imbrication... tout de même...

Registre incorrect
Ce message est envoyé lorsqu'une instruction spécifique du 6809E est définie avec des registres inexistants ou incompatibles entre-eux. Ainsi, les lignes PSHS Z et TFR B,Y provoqueront cette erreur, car Z est un nom de registre inconnu ; de plus, B et Y sont respectivement des registres 8 et 16 bits.

Branchement impossible
Cette erreur signale l'existence d'un branchement relatif dont le point de chute est trop éloigné pour que l'offset de déplacement tienne sur un seul octet.

Syntaxe incorrecte
Ce message intervient lorsque lé champ d'opérande ne correspond pas à la syntaxe générale de la directive. Par exemple, la directive FCC sans apostrophe ou guillemet de tête dans ce champ provoque cette erreur. De même, un champ vide alors que la directive requiert un opérande engendre ce message.

Mode incorrect
Ce message est spécifique aux lignes d'instructions du 6809E par opposition aux directives. Il indique que le champ d'opérande donné ne correspond pas à un mode d'utilisation licite du mnémonique. Ainsi, le mnémonique STU ne peut cohabiter avec l'opérande immédiat #$FCBE.

ANNEXE D

FORMAT INTERNE DE LA TABLE DES SYMBOLES

Nous allons ici décrire la structure interne de la Table des Symboles générée en passe 1 d'un assemblage.
Cette table est constituée par des trinomes indicateur/étiquette/adresse classés par ordre alphanumérique des symboles.

Octet indicateur
Le quartet faible (bits 0, 1, 2, 3) contient la longueur du symbole dans la table. Cette valeur évolue de 1 à 6 unités.
Le quartet fort contient 3 drapeaux servant à la gestion de ces symboles :
bit $20 — redéfinition
- indique qu'une erreur de redéfinition du symbole a été détectée durant l'assemblage. S'il est positionné, le symbole sera précédé d'une étoile dans le listage de la table.

bit $40 — référence
- ce drapeau est positionné lors de la constitution de la table et abaissé en passe 2 dès qu'une référence au symbole suivant a été faite durant l'assemblage. S'il reste positionné à la fin de la passe 2, le symbole sera précédé d'un point d'interrogation au listage de la table.

bit $80 — assignation
- ce bit est également mis à 1 lors de la constitution de la table et abaissé dès qu'une adresse aura pu être calculée et assignée au symbole suivant.

Nom symbolique
Il est codé en ASCII, long de 1 à 6 octets ayant leur bit de parité abaissé.

Adresse
Donnée en deux octets, elle est la valeur représentée symboliquement par l'étiquette. Elle est évaluée en passe 1.