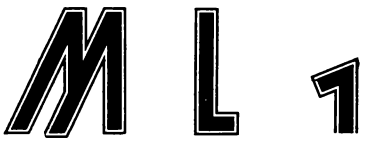

Chargement du programme :
---

Mettre votre LEP en position LECTURE puis tapez RUN". Après ML1 se trouve le programme source "VAR" qui permet de lister toutes les variables activées. Tapez LOAD pour le charger puis @ASS pour générer le code objet. Exec 20000, lance le programme machine.

Ce macro-assembleur en langage machine est destiné à l'ordinateur MO5.

Il a été conçu pour être utilisé conjointement avec le Basic résident de cet ordinateur.

L'introduction des mnémoniques sera donc effectué avec l'éditeur Basic et l'utilisateur peut mélanger librement Basic et langage machine. Il peut même écrire son programme source sans que l'assembleur soit en mémoire.

Avant de poursuivre, signalons que ce mode d'emploi ne constitue pas un cours sur le langage 6809 ni sur les routines internes du MO5. Il est indispensable pour avoir plus de détails, de consulter des ouvrages spécialisés.

## A - UTILISATION DU LOGICIEL

Les mnémoniques seront entrés comme de simples lignes Basic à la différence que celles-ci seront précédées du caractère "'".

Aucun indicateur de début ou de fin du programme source n'est nécessaire.

L'introduction des opérandes (constantes ou adresses absolues) peut se faire en 4 bases différentes :

1) En décimal, en utilisant le préfixe "&".

2) En hexa sans aucun préfixe, mais les nombres doivent être composés littéralement pour un assemblage correct.

3) En octal, en utilisant le préfixe "Q".

4) En binaire, en utilisant le préfixe "%".

Les nombres binaires sans ou avec les zéros non significatifs sont acceptés par l'assembleur.

Exemples : LDX#&255, LDX#00FF, LDX#@117, LDX#%110

Plages de différentes Bases :

| | |
|---|---|
| DECIMAL | 0 à 65535 |
| HEXA | 0 à FFFF |
| OCTAL | 0 à 177777 |
| BINAIRE | 0 à 1111111111111111 |

## C O M M A N D E S

@ASS : Assemblage

@ZERO : Il met à zéro les adresses entre 20000 et 29999 comprises (programmes objet). Cette commande peut servir aussi pour effacer les routines DUMP et DESASSEMBLEUR afin d'avoir le maximum de mémoire disponible pour le code objet. En effet, avec ces routines ENDMEM est 23999 tandis que sans ces routines ENDMEM est 29999.

@SYM : Affichage de la table des symboles.
Les labels sont affichés par groupe de 22.
Pour poursuivre appuyez sur n'importe quelle touche.
A la fin de la table des symboles le message :
" END OF CODE : adresse de fin" est affiché.

@AUT : Cette commande permet l'édition automatique du numéro de ligne suivant, accompagné de l'apostrophe, à chaque pression de la touche ENTREE. Ceci supprime la fastidieuse frappe des numéros devant précéder chaque mnémonique. Pour sortir de ce mode il faut presser CNT-C et vous êtes de retour à l'éditeur habituel du MO5.

@DES : Désassembleur 6809.
Le mode d'emploi est très simple. Après avoir fait @DES, l'écran est effacé et le programme attend l'introduction, en hexadécimal, de l'adresse de départ. Quand le 4ème chiffre est introduit, le MO5 désassemble et affiche instantanément 22 lignes. Pour poursuivre appuyez sur n'importe quelle touche sauf RAZ et ESPACE. La touche

RAZ doit être utilisée pour quitter la commande tandis que la touche d'espacement permet l'introduction d'une nouvelle adresse de départ.

Si pendant l'introduction vous avez commis une erreur de frappe, vous effacez avec RAZ et vous réintroduisez votre adresse.

Les mnémoniques et leur syntaxe sont ceux de Motorola à l'exception de l'adressage étendu indirect où les crochets ont été remplacés par des parenthèses.

Pour faciliter l'exploration des programmes désassemblés les mnémoniques des branchements relatifs sont suivis de l'adresse de branchement.

@ DUMP : Cette commande permet l'exploration rapide de la mémoire de votre ordinateur et la modification immédiate de n'importe quel octet en MEV, simplement en introduisant la valeur hexa désirée.

Pour vous déplacer vous utilisez les flèches, pour introduire une nouvelle adresse de départ vous pressez ENTREE, pour corriger, RAZ, pour quitter la commande, CNT-C.

## INTERACTION BASIC - LANGAGE MACHINE

Deux instructions permettent l'intéraction Basic-langage machine.

* RUN : C'est une macro-instruction de l'assembleur qui permet le chaînage d'exécution dans le sens langage machine-Basic.

↑ X : C'est une nouvelle instruction Basic qui permet de passer un paramètre à un programme assembleur à partir du Basic.

↑ X peut être utilisée indifféremment en mode direct ou dans un programme. En mode programme, elle doit toujours être précédée de ":".

Sa syntaxe est la suivante :

↑ X, valeur à charger dans le registre X, adressse d'exécution, (les virgules sont obligatoires).

Valeur et adresse d'exécution sont des constantes entières positives exprimées en décimal. (Valeurs validés 0 à 65535)

Voici un exemple d'utilisation :

```
10 REM ** Programme Basic
20 :↑X, 65, 20000,:PRINT"B":END
30 REM **Programme Assembleur
40'ORG 4E20
50'TFR X,D
60'PRT
70'RTS
```

Après introduction utilisez la commande ASS pour assembler. Faire RUN, "AB" est affiché.

### B - INSTRUCTIONS 6809

a) Toutes les instructions 6809 en adressage implicite, immédiat, direct, étendu, étendu indirect *1 et indexé sont disponibles.

b) Indexées : Les instructions indexées avec ou sans déplacement *2

et avec ou sans incrémentation, décrémentation sont disponibles. Exemples : LDA,S+, LDB A, Y, LDD D,X, LDA 3,X, LEAY, X++
Les instructions LEAX, LEAY, LEAS, LEAU avec paramètre numérique peuvent également être exprimées par ADDX, ADDY, ADDU, ADDS. Exemples : ADDX#0103, ADDY#&210.

c) Indexées indirectes : Les instructions en adressage indexé indirect sont sans déplacement, ni décrémentation : incrémentation. Exemples de mnémoniques : LDA(,Y), LDB (,S), CMPX(,X), LDA 3,PCR (8 bits).

d) Les PSH et PUL doivent être suivis d'un espace puis de la liste de registres empilés ou dépilés sans espace. Aucun ordre dans l'écriture n'est requis. Exemples : PSHS YXAB, PULU AX.

e) Branchements relatifs : Tous les branchements relatifs courts et longs sont disponibles sauf les branchements nuls : BRN et LBRN.

1 - Déplacement numérique : En complément à deux vers l'arrière. Exemple : BCS FB, BRA &32, LBSR 4E33

2 - Déplacement vers un label. Exemple : LBSR ADR, BRA MOD

*1 Pour les instructions en adressage étendu indirect la syntaxe est différente de celle de Motorola. C'est la directive "!" qui force l'adressage étendu indirect. Exemple : LDA!4E30, LDY!682A.

*2 Le déplacement numérique doit être exprimé en décimal sans préfixe, avec ou sans le signe "+" s'il est positif, avec le signe "-" s'il est négatif.

## INSTRUCTIONS 6809

| | | |
|---|---|---|
| ABX | BITA | COM |
| ADCA | BITB | CWAI |
| ADCB | CLRA | DAA |
| ADDA | CLRB | DECA |
| ADDB | CLR | DECB |
| ANDA | CMPA | DEC |

ANDB
ANDCC
ASLA
ASLB
ASL
ASRA
ASRB
ASR

CMPB
CMPD
CMPS
CMPU
CMPX
CMPY
COMA
COMB

EORA
EORB
EXG R1,R2
INCA
INCB
INC
JMP
JSR

LDA
LDB
LDD
LDS
LDU
LDX
LDY
LEAS
LEAN
LEAX
LEAY
LSLA
LSLB
LSLS
LSRA
LSRB
LSR
MUL
NEGA
NEGB
NEG
NOP
ORA
ORB
ORCC

PSHS
PSHU
PULS
PULU
RORA
RORB
ROLA
ROLB
ROL
ROR
RTI
RTS
SBCA
SBCB
SEX
STA
STB
STD
STS
STU
STX
STY
SUBA
SUBB
SUBD
SWI

SYNC
TFR R1,R2
TSTA
TSTB
TST

Branchements relatifs
---

BCC

BLO

BVC

| | | |
|---|---|---|
| LBCC | LBLO | LBVC |
| BCS | BLS | BVS |
| LBCS | LBLS | LBVS |
| BEQ | BLT | BRN (non |
| LBEQ | LBLT | LBRN disponibles) |
| BGE | BMI | |
| LBGE | LBMI | |
| BGT | BNE | |
| LBGT | LBNE | |
| BHI | BPL | |
| LBHI | LBPL | |
| BHS | BRA | |
| LBHS | LBRA | |
| BLE | BSR | |
| LBLE | LBSR | |

## C - MACRO-INSTRUCTIONS PREDEFINIES

En tout 102 macro-instructions sont disponibles ; 98 en adressage inhérent et 4 en adressage immédiat.
Voici ses instructions en détail avec le nombre d'octets qu'elles occupent et leur fonction.

a) Adressage inhérent

**AAX(5)**
**(X<-X+A)** Le contenu non signé de 8 bits de A est additionné au contenu du registre X. Le résultat est dans X.

Exemple : AAX

| Avant : | Après : |
|---|---|
| X=F000 | X=F0F0 |
| A=F0 | A=F0 |

ADDAB(12)
(D=B+A)  Les accumulateurs A et B sont additionnés et le résultat est dans D (B non signé, A signé)

Exemple : ADDAB

| Avant : | Après : |
|---|---|
| A=FF | A=00 |
| B=FF | B=FE |

ASCA, ASCB(10)
(A ASCII A)  Conversion d'un chiffre dans A (ou B) en ASCII.
<----
Exemple : ASCB

| Avant : | Après : |
|---|---|
| B=0F | B=46 |

BIP(2)

Bip sonore

CLRC(2)
(C<-0)  Met à zéro le bit de retenue

Exemple : CLRC

| Avant : | Après : |
|---|---|
| CC=A5 | CC=A4 |

CLRD(2)
(D<-0)  Met à zéro le registre D

Exemple : CLRD

| Avant : | Après : |
|---|---|
| A=F3 | A=00 |
| B=02 | B=00 |

CLRG(7)
(X,Y,A,B<-0)  Met à zéro les registres A,B,X,Y

CLRH(2)
(H<-0)  Met à zéro l'indicateur H

CLRI(2)
(I<-0)  Met à zéro l'indicateur I

CLRN(2)
(N<-0)  Met à zéro l'indicateur N

CLRU(3)
(U<-0)  Met à zéro le registre U

CLRX(3)
(X<-0)  Met à zéro le registre X

CLRY(4)
(Y<-0)  Met à zéro le registre Y

CLRZ(2)
(Z<-0)  Met à zéro l'indicateur Z

CLRV(2)
(V<-0)  Met à zéro le bit de débordement

CLS(8)
  Efface l'écran. Tous les registres sont sauvegardés

CMPAB(6)
(A-B)  Comparaison directe A et B

CMPXD(6)
(X-D)  Comparaison directe X et D

CMPXY(7)
(X-Y)  Comparaison directe X et Y

CMPYD(7)
(X-D)  Comparaison directe Y et D

COMD(2)
(D-D)  Complémentation logique du D

Exemple : COMD

| Avant : | Après : |
|---|---|
| A=00 | A=FF |
| B=00 | B=FF |

COMX(6)
(X<-X) Complémentation logique du X

COMY(6)
(Y<-Y) Complèmentation logique du Y

D+X(6)
(D<-D+X) Additionne D et X. Le résultat est dans D.

Exemple : D+X

| Avant : | Après : |
|---|---|
| A=A7 | A=A8 |
| B=C0 | B=BF |
| X=00FF | X=00FF |

D-X(6)
(D<-D-X) Le contenu du X est soustrait de l'accumulateur D. Le résultat est dans D.

Exemple : D-X

| Avant : | Après : |
|---|---|
| A=F0 | A=EF |
| B=00 | B=01 |
| X=00FF | X=00FF |

DECD(3)
(D<-D-1) Une unité est soustraite de l'accumulateur D

Exemple : DECD

| Avant : | Après : |
|---|---|
| A=00 | A=00 |
| B=02 | B=01 |

DECS(2)

(S<-S-1)  Une unité est soustraite du registre S

-DECS(2)
(S<-S-2)  Deux unités sont soustraites du registre S

DECU(2)
(U<-U-1)  Une unité est soustraite du registre U

-DECU(2)
(U<-U-2)  Deux unités sont soustraites du registre U

DECX(4)
(X<-X-1)  Une unité est soustraite du registre X

-DECX(4)
(X<-X-2)  Deux unités sont soustraites du registre X

DECY(4)
(Y<-Y-1)  Une unité est soustraite du registre Y

-DECY(4)
(Y<-Y-2)  Deux unités sont soustraites du registre Y

HLA,HLB(11)
(0 2 <- 2 0)  Echange quartet haut et bas du A et B
( A  A )

Exemple : HLA

| Avant : | Après : |
|---|---|
| A=F0 | A=0F |

HLD(2)

( F 0 1 2 <- 1 2 F 0 )  Echange octet fort, octet faible du D
(  D  D  )

HLU,HLX,HLY(6)

```
( 0 1 2 1  <-  2 1 0 1 )        Echange octet fort, octet faible du U,X
(    X           X    )        ou Y
```

Exemple : HLU

| | Avant : | Après : |
|---|---|---|
| | U=0F34 | U=340F |

INCD(3)
(D<-D+1)  Un 1 est ajouté à l'accumulateur D

Exemple : INCD

| | Avant : | Après : |
|---|---|---|
| | A=FF | A=0 |
| | B=FF | B=0 |

INCS(2)
(S<-S+1)  Un 1 est aouté au registre S

+INCS(2)
(S<-S+2)  Un 2 est ajouté au registre S]

INCU(2)
(U<-U+1)  Un 1 est ajouté à U

+INCU(2)
(U<-U+2)  Un 2 est ajouté à U

INCX(2)
(X<-X+1)  Un 1 est ajouté à X

+INCX(2)
(X<-X+2)  Un 2 est ajouté à X

INCY(2)
(Y<-Y+1)  Un 1 est ajouté à Y

+INCY(2)
(Y<-Y+2)  Un 2 est ajouté à Y

INIT(16)

Initialise un bloc avec le contenu de l'accumulateur. A. Début du bloc en X, fin du bloc en Y.

JMPX,JMPY(3)
(PC<-X) Autre forme de mnémonique pour JMP,X ou JMP,Y

KEY(5)

Comme INPUT$(1) du basic. Code ASCII de la touche pressée dans l'accumulateur B

LDXAB(6)
(X<-B+A) Les accumulateurs A et B sont additionnés et le résultat est dans X (B non signé, A signé)

Exemple :LDXAB

| Avant : | Après : |
|---|---|
| X=F343 | X=0000 |
| A=FF | A=FF |
| B=01 | B=01 |

LDYAB(8)
(Y<-A+B) Les contenus signés des accumulateurs A et B sont additionnés. Le résultat est dans Y.

Exemple : LDYAB

| Avant : | Après : |
|---|---|
| Y=0000 | Y=FFFE |
| A=FF | A=FF |
| B=FF | B=FF |

MOVE(29)

Déplace bloc 1 --> bloc 2. Adresse bloc 1 en X, bloc 2 en Y, nombre d'éléments en D. Tous les registres (sauf CC) sont sauvegardés.

POP(2)
(S=S+2) Dépile une adresse de retour

PRT(2)

Affiche le caractère ASCII dont le code est en B. Tous les registres sont sauvegardés.

Exemple : LDB#41
PRT Affiche la lettre A.

RESET(3)
(PC<-F000) Initialise le processeur. Cette instruction doit être employée uniquement dans le cas de nécessité absolue. L'intégrité du système n'est pas garantie après l'utilisation de la fonction RESET.

RET (Groupe RTS Conditionnels)(3)

RET CC (RTS on Carry Clear. Retour si retenue à 0)
RET CS (RTS on Carry Set. Retour si retenue à 1)
RET EQ (RTS on Equal. Retour si égal)

RET GE (RTS on Greater than or Equal to. Retour si supérieur ou égal à )
RET GT (RTS on Greater than. Retour si supérieur à)
RET HI (RTS on Higher. Retour si supérieur)
RET HS (RTS on Higher or Same. Retour si supérieur ou égal)
RET LE (RTS on Less than or Equal to. Retour si inférieur ou égal à)
RET LO (RTS on Lower. Retour si inférieur)
RET LS (RTS on Lower or same. Retour si inférieur ou égal)
RET LT (RTS on Less than. Retour si inférieur à)
RET NC (RTS on N Clear. Retour si N=0)
RET NE (RTS on Not Equal. Retour si pas égal)
RET NS (RTS on N Set. Retour si N=1)
RET NZ (RTS on Not Z. Retour si Z=0)
RET VC (RTS on Overflow Clear. Retour si V=0)
RET VS (RTS on Overflow set. Retour si V=1)
RET Z (RTS on Z. Retour si Z=1)

ROLD(2)

( b15 D bo ) Rotation à gauche du D
( C )

Exemple : ROLD

```
            Avant :          Après :
            A=F0             A=E0
            B=00             B=00
         bit C=0
```

ROLX,ROLY,ROLU(6)
Rotation à gauche du X,Y,U.

RORD(2)

```
(  b15      bo  )
(       C       )
```

Rotation à droite du D

Exemple : RORD

```
            Avant :          Après :
            A=FF             A=FF
            B=FF             B=FF
         bit C=0
```

RORU,RORX,RORY(6)
Rotation à droite de U,X,Y

SETC(2)
Met à 1 le bit de retenue

SETH(2)
Met à 1 l'indicateur H

SETI(2)
Met à 1 l'indicateur I

SETN(2)
Met à 1 l'indicateur N

SETV(2)
Met à 1 l'indicateur V

SETZ(2)
Met à 1 l'indicateur Z

WAIT(13)

Temporisation de 0 à FFFF. Durée en X

X+D(10)
(X<-X+D)  Additionne D+X. Résultat en X.

X-D(10)
(X<-X-D)  Le contenu de D est soustrait du registre X. Le résultat est dans X.

ZERO(16)

Il met à zéro un bloc dont le début est en X et la fin en Y.

*RUN(3)
(PC<-C54B)  Il permet de chaîner l'exécution dans le sens assembleur --> basic.

b) Adressage Immédiat

ADDX(2+2)
(X<-X+M)  L'opérande de 16 bits est additionnée avec le contenu du registre X. Résultat en X.

Exemple : ADDX#00F0

| Avant : | Après : |
|---|---|
| X=0003 | X=00F3 |

ADDU,ADDS,ADDY(2+2)

L'opérande de 16 bits est additionnée avec le contenu du registre U,S,Y. Le résultat est dans le registre spécifié.

Exemple : ADDY#848

| Avant : | Après : |
|---|---|
| Y=FFFF | Y=002F |

c) Toutes les instructions en adressage immédiat, étendu et étendu indirect, peuvent être paramétrées avec un symbole. Par exemple LDX#@MAX chargera dans X l'adresse pointée par MAX, tandis que LDY>@MAX chargera dans Y le contenu de l'adresse pointée par MAX. Cette possibilité permet en combinaison avec l'adressage indexé d'accéder facilement à un tableau.

Voici un exemple :

```
0'      LDA>@MAX
1'      LDB>@MIN
2'      LDX>@ADR
3'      RTS
4'@MAX  FCB FF
5'@MIN  FCB 01
6'@ADR  FDB 5DC0
```

Après exécution A contient FF, B, 01 et X, 5DC0.

d) Quatre instructions en adressage immédiat : LDA, LDB, CMPA, CMPB peuvent être paramétrées directement avec un caractère ASCII affichable. Pour ce faire elles doivent être préfixées avec "A".

Exemple : ALDA#B stocke la lettre B dans l'accumulateur A.

## D - MACROS DEFINIES PAR L'UTILISATEUR

L'utilisateur peut définir des macro qui seront appelées par le label de la macro précédé de 2 points (:). Exemple ; : @MAX éxecutera la macro-instruction @MAX. La définition d'une ou plusieurs MACRO doit commencer par la directive MACRO et finir par la directive ENDM. *1

Source : Chaque "MACRO-UTILISATEUR" doit commencer par un label et finir par un RTS. *2

Elle peut contenir n'importe quelle instruction ou directive (sauf

ORG) et elle peut même appeler à son tour une autre macro. En cas d'utilisation des macros sachez que les adresses 23000 à 23999 sont réservées.

*1 NOTE 1 : L'absence de la directive ENDM ne provoque pas d'erreur d'assemblage, mais vous devez l'utiliser systèmatiquement pour éviter une éventuelle erreur à l'exécution du programme objet.

*2 NOTE 2 : Les Macro ne sont exécutables que par l'appel ":label", sinon elles sont ignorées à l'exécution du programme objet.

E - LABELS

Les labels doivent être composés impérativement de 3 caractères. Ils doivent commencer par l'indicateur @.

Entre le label et l'instruction de la même ligne un espace est obligatoire.

Il n'y a aucune restriction quant aux caractères à utiliser pour composer les labels. D'autre part il n'existe pas de mots réservés. Les opérations autorisées sur les labels sont l'addition et la soustraction (Bases admises : Dec,Bin,oct).
Exemple :@LDX#@AAA+&99 ; LDA>@AAA-%110

F - DIRECTIVES
---

Les paramètres de toutes les directives d'assemblage doivent être exprimés en hexa.

ORG : La directive ORG charge le PC à l'adresse spécifiée par le paramètre. Les instructions qui suivent ORG sont assemblées en incrémentant cette adresse. Il peut y avoir plusieurs ORG dans un programme source. Si ORG est omise dans le programme source, le PC est chargé à 20000. Les adresses valides pour ORG sont comprises entre 20000 et 29000.
Exemple : ORG 5E24. Le début du programme objet sera à l'adresse 24100.

INSERT ADR1, ADR2 :

La directive INSERT insére, dans des positions mémoire définies par la valeur courante du PC, le code objet qui commence à l'adresse ADR1 et fini à l'adresse ADR2.
Exemple : INSERT 4E20, 4E30 insére les données ou le programme en langage machine qui commence à l'adresse 4E20 et fini à 4E30.

RMB (RESERVE MEMORY BYTES) :

La directive RMB réserve un espace mémoire non initialisé à une valeur particulière.
Exemple : RMB OF, réserve 15 octets.

FCB (FORM CONSTANT BYTE) :

FCB stocke des constantes 8 bits, exprimées en hexa, dans des positions mémoire définies par la valeur courante du PC. FCB peut définir plusieurs constantes séparées par une virgule.
Exemple : FCB OF,FF,01.

FDB (FORM DOUBLE BYTE) :

FDB stocke des constantes 16 bits, exprimées en hexa, dans des positions mémoire définies par la valeur courante du PC. FDB peut définir plusieurs constantes 16 bits.
Exemple :FDB FFFF, 4EFF, 00F3.

=(EQUATE) :

La directive = attribue à l'étiquette désignée une valeur de 16 bits exprimée en hexa. Cette valeur n'est pas liée au compteur programme. Les symboles définis avec = ne peuvent pas être redéfinis dans la suite du programme.
Exemple : @MAX=1F40

$ : Cette directive stocke des chaînes de caractères dans des positions mémoire définies par la valeur courante du PC.
Exemple : $ORDINATEUR MO5.

END : A la rencontre de cette directive optionnelle l'assemblage s'arrête. Les instructions qui suivent END sont ignorées. L'utilisateur peut utiliser une étiquette avec END. Répétons-le,

cette directive n'est en aucun cas obligatoire et peut ne pas être présente.

MACRO et ENDM : Début et fin de définition d'une ou plusieurs macros.

BRK : L'insertion de cette directive aux endroits désirés dans le programme source permet l'exécution du programme pas à pas avec visualisation des registres internes. A chaque arrêt la valeur courante de tous les registres (PC, X, Y, A, B, U, S, CC, DP) est affichée. Pour poursuivre appuyez sur n'importe quelle touche.

## G - INTERRUPTIONS SOFTWARE

Vous pouvez utiliser des SWI directement suivis du code d'entrée au moniteur du M05.

```
Exemple : SWI
          FCB 02
```

Les codes, 58 en tout, sont répartis en deux groupes différenciés par le contenu du bit 7. Si le bit 7 est à 0, un "JSR" à la routine est simulé, s'il est à 1, un "JMP" est simulé.

## H - ERREURS

SN ERROR : Erreur de syntaxe ou de paramètre.

ERROR MEM : a) La mémoire pour l'implantation du programme objet est insuffisante.

ou b) Il n'y a plus de place dans la table des étiquettes.

ou c) Il n'y a plus de place pour la définition d'une macro.

DUPLICATE ERROR : Un même label est utilisé plus d'une fois.

BAD LABEL : Branchement à un label inexistant

BRANCH OUT OF RANGE : L'adresse d'un branchement est trop éloignée de l'adresse de l'instruction courante.

BAD ADDRESS : L'adresse indiquée pour l'implantation du code objet est incorrecte. Chaque erreur détectée arrête l'assemblage. Le No de la ligne erronée est affichée ainsi que la ligne même permettant ainsi une correction immédiate.

## I - PROGRAMME OBJET et SOURCE

L'assembleur occupe les adresses 30000 à 40959.
Sans ORG le programme objet est implanté à partir de l'adresse 20000. ENDMEM est 29999. (23999 avec DUMP et DESASSEMBLEUR en mémoire)

Les programmes Basic et (ou) source peuvent occuper la MEV jusqu'à l'adresse 19999.

Pour exécuter le programme objet il faut faire EXEC (adresse).

## J - SAUVEGARDE ET CHARGEMENT DU PROGRAMME SOURCE ET OBJET

Pour sauvegarder sur cassette le programme source courant il faut faire SAVE "NOM", comme pour un programme Basic normal.

Le programme objet est sauvegardé par SAVEM "NOM", adresse de début, adresse de fin, adresse de début. L'utilisateur peut choisir le suffixe pour l'enregistrement. Par exemple SAVE "PRG.ASS" sauvegardera le programme source avec comme type de fichier "ASS".

Pour avoir l'adresse de fin vous pouvez faire : PRINT PEEK (40949) *256 + PEEK (40950)-1. L'adresse de fin est également donnée par la commande @SYM.

Pour charger un programme source depuis une cassette il faut utiliser LOAD, comme pour un programme Basic. Pour charger un programme objet il faut utiliser l'instruction LOADM.

## K - DUREE DE L'ASSEMBLAGE

| | |
|---|---|
| 100 lignes | 1,5" à 3" |
| 1000 lignes | 19" à 30" |

# A N N E X E

1. En cas de sauvegarde sur cassette d'un code objet contenant des appels à des macro-instructions définies par l'utilisateur (adresses 23000 à 23999) doit être sauvegardée conjointement pour une éxecution ultérieure correcte.
   Les macros-utilisateur peuvent également être sauvegardées séparement par SAVEM "NOM.MAC", 23000, 23999, 23000.

2. La directive BRK génère un JSR (BD 77EC) à une routine interne du ML1. Elle ne doit donc être utilisée qu'avec ML1 en mémoire.

3. La constante avant la virgule dans le monde déplacement PCR est un offset et non pas une adresse effective.

   Exemple :

| | ASSEMBLEUR | ADRESSE | CODES GENERES |
|---|---|---|---|
| | LEAX 2,PCR | 4E20 | 308C02 |
| | BRA @SUI | 4E23 | 2004 |
| | FDB F000,AB00 | 4E25 | F000AB00 |
| SUI | LDB,X | 4E29 | E684 |
| | - | | |
| | - | | |
| | - | | |

   X sera chargé avec 4E25.

ML1 est un macro-assembleur symbolique qui contrairement aux assembleurs conventionnels permet de mixer Basic et assembleur à volonté.

Editeur plein écran, 102 macro-instructions prédéfinies qui rendent la programmation en langage machine accessible aux débutants, désassembleur 6809, numérotation automatique, moniteur hexadécimal pour travailler directement sur la mémoire du MO5, définition des macros par l'utilisateur, chaînage d'éxecution dans le sens assembleur-basic, passage de paramètres à partir du Basic.