P 3 0 R S 0 9

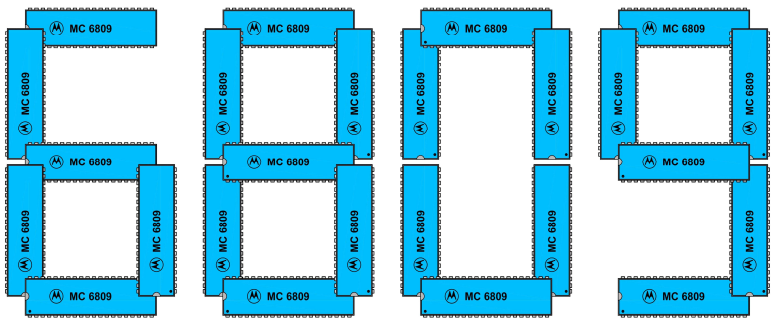
version 2.13

DOCUMENTATION

UTILISATEUR DU PROGRAMME

ASSEMBLEUR - DESASSEMBLEUR

Pour le microprocesseur



Richard SOREK **keros6809@gmail.com**

Révision de ce document le 15/11/2024 00:03

En premier, je vous demanderai d'avoir la gentillesse de m'aider à améliorer les textes de cette documentation (incompréhension, fautes d'orthographe, etc ...) et de faire remonter des éventuels problèmes rencontrés ainsi que vos remarques.

Dans de cas vous pouvez me joindre à l'adresse mail **keros6809@gmail.com**
Ne pas oublier de me préciser le numéro de la version de votre P30RS09

Pourriez-vous également m'aider en me faisant part d'éventuels BUGs que vous rencontreriez.
Dans ce cas j'aurai besoin d'une copie du fichier sur lequel vous travaillez afin que je puisse le faire dérouler pas à pas dans mon compilateur.

Pour votre information, j'ai créé plusieurs documentations très détaillées sur 6809, 6821, 6850 et 6840.
Cette documentation se présente sous la forme de fichiers PDF.

Ces ouvrages de synthèse sur le microprocesseur 6809 (Tome 1) et de ses périphériques (Tome 2), sont le fruit de plusieurs années de travail. La mise en page et la création de croquis réalisé durant mes soirées, mes nuits d'insomnies, mes week-ends et mes vacances.

Dans cet ouvrage (sans exception) :
Tous les textes ont été saisis...
Tous les tableaux ont été créés...
Tous les croquis ont été dessinés... par mes propres soins.

Ceci représente de très nombreuses heures, de très nombreux jours au service de ces documents de travail, ces derniers ayant pour seule ambition, d'être des guides de référence gratuit pour tous les passionnés du 6809 et de ses périphériques.

Je peux vous faire parvenir ces ouvrages gracieusement, pour ce faire envoyez moi un mail à l'adresse **keros6809@gmail.com**

Je ferais parvenir par mail, les mises à jour de ces documents et mon l'ASSEMBLEUR-DESASSEMBLEUR le P30RS09 aux personnes qui m'auront fait l'honneur de me communiquer leur mail.

N'hésitez pas à me contacter pour savoir si il n'y a pas des versions plus récente.

L'assembleur étant un langage assez rigoureux, j'ai souhaité que les versions de mon ASSEMBLEUR-DESASSEMBLEUR le P30RS09 soit beaucoup moins permissif pour l'assemblage que certain logiciels, notamment ceux provenant de certains constructeurs Américain.

Pour assembler vos programmes sources à l'aide du P30RS09, il est nécessaire de :
1) Enregistrer le fichier en code AINSI pour s'échapper du code UNICODE.
2) Remplacer les astérisques par des points-virgules pour les commentaires.
3) Mettre des points-virgules devant chaque commentaires.
Le **";"** étant le seul et unique caractère reconnu comme le début des commentaires dans le P30RS09 (l'astérisque "*" étant réservé aux opérations et expressions mathématiques).

Je reste à votre écoute et vous souhaite un bon courage et une bonne utilisation de ce programme.

Richard SOREK **keros6809@gmail.com** (59169 Férin - Nord de la FRANCE)

PS : Veuillez être indulgent, je ne suis par programmeur de formation, je vous demande simplement avoir votre retour pour améliorer mon programme ou mes documentations, merci !

SOMMAIRE GENERAL

A : IMPERATIF : A faire avant l'utilisation du P30RS09

1. **CHAMP ETIQUETTES :**
   Pour les étiquettes et pour les noms des constantes déclarées par EQU :
   - Le premier caractère de ce champ ne doit pas être numérique.
   - Ce champ ne doit pas être entièrement numérique
   - Ce champ ne doit pas correspondre à :
     - un nom de registre, tel que **PC, PCR, Y, X, S, U, DP, A, B, D ou CC**
     - un nom d'un mnémonique
     - un nom d'une directive d'assemblage
   - Les caractères autorisés sont :
     0.....à .....**9** (sauf pour le premier caractère)
     a.....à .....**z**
     A.....à .....**Z**
     et les caractères suivant **é è à . _ | : =**.

2. **POUR LES COMMENTAIRES :**
   Le seul caractère qui permet la désignation d'un commentaire est le point-virgule **;** ($3B)
   L'astérisque ***** ($2A) est réservé aux formules mathématiques dans les opérandes et à la désignation de l'adresse courante.

   Impératif :
   - Avant de retravailler un listing, qui n'a pas été assemblé par le **P30RS09**, il faut remplacer les ***** devant les commentaires de bloc (commentaires commençant dans par la colonne étiquette) par des **;**
   - Mettre un ESPACE devant le ; pour bien séparer l'opérande du commentaire.

3. **AVANT L'ASSEMBLAGE :**
   Avant l'assemblage d'un listing qui n'a pas encore été assemblé par le **P30RS09** il faudra :
   - Pour les tableaux des SYMBOLES en fin de listing
     - Soit supprimer les lignes en fin de fichier concernant ces tableaux
     - Soit mettre un point-virgule devant chaque ligne, pour placer ces tableaux en commentaires.
     - Soit mettre le caractère "|" code ASCII 124 ($7C) devant chaque ligne, pour que ces dernières soient invisibles dans le listing de sortie.
   - Remplacement de tous caractères **TAB** (tabulation $09) par un caractère **SPACE** $20.
   - Ne pas oublier de remplacer les ***** devant les commentaires de bloc par des **;**
   - Si le listing a déjà été assemblé ultérieurement, voir si il y a des numéros de lignes, sinon il faut en ajouter (En passant par exemple par un tableur puis dans un traitement de texte pour remplacer les caractères **TAB** par des **SPACE**.

4. **En partant d'un listing résultant d'un assemblage antérieur**
   Si il y a une adjonction d'une ligne **EQU** en tout début du listing, alors **il est impératif que cette nouvelle ligne soit au même format que les lignes suivantes.**

   Sinon le programme **P30RS09** n'arrivera pas a déterminer exactement le format et le début des informations à traiter en vue d'un nouvel assemblage (Etiquettes, Mnémoniques, Opérandes et Commentaires)

B : Autres Renseignements sur le fonctionnement du P30RS09

1. Le **P30RS09** en tout début d'assemblage va détecter le format du fichier à traiter, pour cela il va rechercher dans les lignes valides la première ligne avec la directive EQU et il va déterminer la position du premier caractère de la zone étiquette.

2. Si on doit ajouter des lignes dans un fichier qui a déjà été assemblé :
   - Soit on respecte le format en cours du fichier
     (colonne étiquette, colonne mnémonique, colonne opérande, colonne commentaire)
   - Soit on ne veut pas respecter le format du fichier gagner du temps alors :
     - Pour une ligne avec étiquette, on place l'étiquette en première colonne.
       Le mnémonique, l'opérande et le commentaire suivent espacés d'un caractère SPACE
     - Pour une ligne sans étiquette on place le début du mnémonique en 2ième colonne

3. Possibilité de mettre deux étiquettes la même adresse (comme dans certain listing US).
   Le format est le suivant

```
Etiquette_01                              ; ligne sans Mnémonique
Etiquette_02        STA     $FF           ;
                    LDB     . . . . .     ;
```

4. Après l'assemblage par le **P30RS09** :
   - Le fichier de sortie garde le même nom, mais une copie de sauvegarde de l'ancienne version est réalisée.

     Cette copie a comme suffixe `[P30RS09 vX.XX Version-du-AAMMJJ_HHMM].txt`
     Avec `AAMMJJ_HHMM` correspondant à la date et l'heure de l'ancien fichier qui a été assemblé.

   - Création de plusieurs fichiers de sortie, dans le même répertoire que le fichier d'entrée.
     Ces derniers reprennent le même nom que le fichier en entrée mais avec les suffixes suivants :

`[P30RS09 vX.XX Assemblage].txt`	Pour le résultat de l'assemblage
`[P30RS09 vX.XX Code-Objet].txt`	Pour le code Objet (code machine exécutable)
`[P30RS09 vX.XX Tab-Constantes].txt`	Tableau des Constantes **si l'option D est positionnée** (possibilité de l'intégrer dans un fichier EXCEL)
`[P30RS09 vX.XX Tab-Etiquettes].txt`	Tableau des Etiquettes **si l'option D est positionnée** (possibilité de l'intégrer dans un fichier EXCEL)

5. AUTRES DIRECTIVES D'ASSEMBLAGE :

   Pour l'instant le **P30RS09** ne gère pas les directives d'assemblage suivantes :
   BSZ, FAIL, FILL, NAM, OPT, PAGE, REG, SET, SPC, TTL, ZMB

   Si vous avez des informations concernant ces directives alors contactez moi à **keros6809@gmail.com**

6. Cas de l'adressage Indexé avec PCR

   Il est mis dans le commentaire, la valeur décimale et hexa de l'opérande sous la forme :

   {{Val Opérande = xxxx soit $yyyy}}

   Au début il y a deux parenthèses ouvertes **{ $7B**
   A la fin il y a deux parenthèses fermées **} $7D**

C : Directives FCB et FDB

1. Uniquement pour les directives FCB et FDB (pas pour FCC), il y a la possibilité de mettre une donnée supplémentaire en mettant deux virgules côte à côte. Exemples :
   `Etiquette  FCB  $F4,'A,,FF  ;` on place la valeur $00 dans la troisième donnée.

2. Dans les listings à assembler, les lignes secondaires (contenant uniquement le reste des codes hexa) :
   - **Il est impératif** de ne pas mettre les numéros de ligne devant l'adresse.
   - Ne pas mettre de commentaires dans ces lignes, car ils seront ignorés dans le listing final.

D : Directive FCC

1. Pour FCC, il est impératif de désigner le début du champ commentaire par la présence d'un point virgule en début du commentaire.

2. Pour cette directive, seul les caractères suivants peuvent être utilisés comme délimiteurs :
   `"` Valeur Hexa $22
   `|` Valeur Hexa $7C

3. Quelques exemples de déclaration :

```
FCC   "Texte"     ; chaine de caractère
FCC   NomVar      ; nom de constante (directive EQU)
FCC   'A          ; caractère ASCII
FCC   1234        ; donnée Décimale      (de 0 à 65 535)
FCC   $F3         ; $  donnée Héxa       (de 0 à FFFF)
FCC   @1177       ; @  donnée Octal      (de 0 à 17 7777)
FCC   %100111     ; %  donnée Binaire    (de 0 à 1111  11111  1111  1111)
```

4. Pour la directive FCC : 10 données maximum par ligne, séparées par des virgules.

5. Pour répéter plusieurs fois le même caractère on utilise le format `(x/'c')` ou `(xx/'c')`
   avec :
   - `x` ou `xx` un nombre en décimal sur 2 caractères maxi (1 à 99)
   - `/` le séparateur
   - `' '` les simples côtes, encapsulant le caractère à répéter
   - `C` le caractère ASCII a répéter dans les données de FCC, sauf le caractère `'` ($27)

   exemple : `FCC  (10/'-'),"RESULTAT",(10/'-')`

   donne : `----------RESULTAT----------`

6. Si on souhaite mettre dans le texte :
   -- Un point-virgule il faudra mettre le code $3B `FCC  "Exemple",$3B," pour le..."`
   -- Une virgule il faudra mettre le code $2C `FCC  "Monter le son",$2C," dans la..."`

E : Directive ORG

1. Zone d'octets non programmés en tout début de listing
   Une zone de saisie appelée **[Adresse Début d'Assemblage]** est prévue pour indiquer à l'Assembleur à quel endroit de l'espace mémoire doit on incorporer le début du code objet.

   Si cette zone est renseignée, elle doit impérativement être sur 4 caractères et en Hexa sans le signe $.

   Si dans le listing à assembler, la première ligne ORG donne une adresse plus grande que la zone de saisie [Adresse Début d'Assemblage], alors il y a insertion de code $FF (octets non programmés) devant le code d'assemblage de la première instruction.

2. Zone d'octets non programmés à l'intérieur du listing
   Dans le cas où une ligne ORG serait à l'intérieure du listing, alors dans le code objet, on comble les adresses non occupées par des $FF.

   Par contre si l'adresse de la ligne ORG aurait une valeur plus petite que la dernière adresse libre alors il y aura une erreur.

   Si ces deux adresses sont égales, alors il y aura insertion dans le commentaire le texte :
   " <<<<<<<<<<<<<<<<<< Cette ligne ORG est inutile !".

F : Fichier P30RS09.INI

1. Le fichier P30RS09.ini se trouve normalement dans **C:\windows**
   A défaut il sera dans un des autres disque C: D: E: ou F: En fait ce sera le disque ou l'on trouvera le répertoire Windows
   C'est un fichier texte modifiable par le Bloc-Note.

2. **Il est impératif** de respecter certaines règles :
   - Les informations débutent juste après le signe = ne pas mettre d'espace entre les deux
   - Le **]** doit être juste devant le signe **=** ……]=…..
   - Tous les champs ci-dessous, doivent être présents dans le fichier .ini
   - Le texte entre **[** et **]** doit être rigoureusement identique à ci-dessous

3. **Les divers champs sont :**

```
[Répertoire fichier ini]=C:\WINDOWS\P30RS09.ini
[Lecteur courant]=\\tsclient\C
[Répertoire pour l'Assembleur]=\\tsclient\C\Richard……
[Répertoire pour le DéAssembleur]=\\tsclient\C
[Réserve 01]=
[Réserve 02]=
[Répertoire pour la Visualisation de fichier en Hexa]=\\tsclient\C
[Répertoire pour le traitement S1S9]=\\tsclient\C
[Répertoire pour les Traitements sur Fichiers]=\\tsclient\C\Richard……
[Choix de la taille de l'étiquette]=20
[Choix de la taille de l'opérande]=20
```

G : Options de la fenêtre Assemblage

Option A AVEC (ou sans) Recherche d'optimisation. 
Lors de l'assemblage le **P30RS09** va déterminer :
- Si certains branchements peuvent être passés de 16 en 8 bits.
- Si l'on peut passer de l'adressage Etendu vers l'adressage Direct.

Le **P30RS09** indiquera ces possibilités par l'apparition d'une erreur.

Option B SANS (ou avec) les tableaux des Symboles (Etiquettes et Constantes) en fin de listing. 
Pour les gros programmes sources, le P30RS09, peut mettre quelque temps pour trier les tableaux des symboles, pour gagner du temps lors du développement et de la mise au point, ne pas sélectionner pas cette case.

Option C SANS (ou avec) les adresses de branchements dans le listing : 
Avant le champ Etiquette et pour chaque ligne concernée par un branchement, il y a possibilité de supprimer les adresses de branchements. Ceci peut alléger la lecture du programme source en terme de visibilité.

Option D AVEC (ou sans) Création des Fichiers des Constantes et des Etiquettes. 
Avec cette option le P30RS09 va créer deux fichiers dans le même répertoire, avec le même nom mais complémenté des suffixes suivants :

Pour l'un, contenant toutes les Constantes : **[P30RS09 vX.XX Tab-Constantes].txt**
Pour l'autre, contenant toutes les Etiquettes : **[P30RS09 vX.XX Tab-Etiquettes].txt**

Ces deux fichiers peuvent très facilement, par un copier coller de leur contenu, intégrer un tableur en vue d'un autre traitement.

Option E SANS (ou avec) Les commentaires à la fin du Listing. 
Cette option permet d'arrêter le listing juste après la dernière instruction assembleur.
Il n'y a donc pas d'impression des divers renseignements sur l'assemblage

Option F AVEC (ou sans) continuité des adresses dans le code objet. 
Pour cette option, si elle est cochée, il apparaît un champ de saisie :

Adresse de Début d'Assemblage

Exemple SANS continuité

On aura le code objet comme ceci

```
54C0AF53108F54218F55D09F59A0BF58D0CF5A91CF5AE07F5B41BF5C95510246
7582378505620043001310332053307340935 0B360D370F3D3D056500011B200
0674000196000004144522020 4D45204B370A0D04

Code Objet à implanter à l'adresse  $FFF0
----------------------------------------
FE9BFE9FFEA3FEA7FEABFEAFFEB3F036
```

Exemple AVEC continuité

On aura le code objet comme ceci (les espaces non programmé entre deux adresses seront comblés par des FF.

```
54C0AF53108F54218F55D09F59A0BF58D0CF5A91CF5AE07F5B41BF5C95
5102467582378505620043001310332053307340935 0B360D370F3D3D0
56500011B200067400019600000414452202 04D45204B370A0D04FFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE9BFE9FFEA3FEA7FEABFEAFF
EB3F036
```

Dans le listing d'assemblage il peut y avoir des erreurs A314, ceci se produit lorsque l'adresse courante est inférieure à l'adresse de la ligne précédente.

Exemple : erreur à la ligne 00210

```
A314 à l'adresse $xxxx : L'adresse de la ligne courante 00210 est
      inférieure à l'adresse de la ligne précédente.
|            Adresse calculée de la ligne précédente = $F000
|                          Adresse de ligne courante = $EFFC
|IMPORTANT : Vérifier si il n'y a pas du code derrière cette
|            adresse, sinon le code objet serai faussé !
```

Après un code A314 il faut vérifier si il n'y a pas du code après la ligne en erreur, dans ce cas la continuité dans les adresses n'est pas respectée dans le code objet.

Option G SANS Numéros Ligne dans le Listing d'Assemblage. 

Permet de supprimer la zone du numéro de ligne, la zone Adresse est alors en colonne une.

Option H AVEC valeur du PCR dans les commentaires. 

Pour l'adressage Indexé on ajoute dans le commentaire la valeur décimale de l'opérande bordée par des doubles parenthèses **{{.....}}**

exemple : **{{Val Opérande = -4139 soit $EFD5}}**

H : Formats des Expressions Arithmétiques pris en compte pour le calcul dans les Opérandes

Avec **AAA, BBB, CCC ou DDD** pouvant être un mélange :
d'Etiquette
de Constante
de Variable Décimale, Hexa, Octal, Binaire ou ASCII

* = adresse courante

```
§RF100      *

§RF110      *+BBB
§RF111      *-BBB
§RF112      AAA+*    ou    -AAA+*
§RF113      AAA-*    ou    -AAA-*

§RF120      *+BBB+CCC
§RF121      *-BBB-CCC
§RF122      *+BBB-CCC
§RF123      *-BBB+CCC

§RF130      AAA+BBB+*    ou    -AAA+BBB+*
§RF131      AAA-BBB-*    ou    -AAA-BBB-*
§RF132      AAA+BBB-*    ou    -AAA+BBB-*
§RF133      AAA-BBB+*    ou    -AAA-BBB+*

§RF140      (AAA*BBB)+*    ou    -(AAA*BBB)+*
§RF141      (AAA*BBB)-*    ou    -(AAA*BBB)-*

§RF150      *+(BBB*CCC)
§RF151      *-(BBB*CCC)

§RF160      (*+BBB)*CCC    ou    -(*+BBB)*CCC
§RF161      (*-BBB)*CCC    ou    -(*-BBB)*CCC

§RF170      (*+BBB)/CCC    ou    -(*+BBB)/CCC
§RF171      (*-BBB)/CCC    ou    -(*-BBB)/CCC

§RF180      (*+BBB)/CCC+DDD    ou    -(*+BBB)/CCC+DDD
§RF181      (*-BBB)/CCC-DDD    ou    -(*-BBB)/CCC-DDD
§RF182      (*+BBB)/CCC-DDD    ou    -(*+BBB)/CCC-DDD
§RF183      (*-BBB)/CCC+DDD    ou    -(*-BBB)/CCC+DDD

§RF190      (*+BBB)*CCC+DDD    ou    -(*+BBB)*CCC+DDD
§RF191      (*-BBB)*CCC-DDD    ou    -(*-BBB)*CCC-DDD
§RF192      (*+BBB)*CCC-DDD    ou    -(*+BBB)*CCC-DDD
§RF193      (*-BBB)*CCC+DDD    ou    -(*-BBB)*CCC+DDD
```

Ne faisant pas intervenir l'adresse courante

```
§RF200      AAA     ou     -AAA

§RF210      AAA+BBB     ou     -AAA+BBB
§RF211      AAA-BBB     ou     -AAA-BBB
§RF212      AAA*BBB     ou     -AAA*BBB
§RF213      AAA/BBB     ou     -AAA/BBB

§RF220      AAA+BBB+CCC     ou     -AAA+BBB+CCC
§RF221      AAA-BBB-CCC     ou     -AAA-BBB-CCC
§RF222      AAA+BBB-CCC     ou     -AAA+BBB-CCC
§RF223      AAA-BBB+CCC     ou     -AAA-BBB+CCC

§RF230      (AAA+BBB)*CCC     ou     -(AAA+BBB)*CCC      (idem §RF270 avec DDD=0)
§RF231      (AAA-BBB)*CCC     ou     -(AAA-BBB)*CCC      (idem §RF271 avec DDD=0)
§RF232      (AAA+BBB)/CCC     ou     -(AAA+BBB)/CCC      (idem §RF280 avec DDD=0)
§RF233      (AAA-BBB)/CCC     ou     -(AAA-BBB)/CCC      (idem §RF281 avec DDD=0)

§RF240      (AAA*BBB)+CCC     ou     -(AAA*BBB)+CCC
§RF241      (AAA*BBB)-CCC     ou     -(AAA*BBB)-CCC
§RF242      (AAA/BBB)+CCC     ou     -(AAA/BBB)+CCC
§RF243      (AAA/BBB)-CCC     ou     -(AAA/BBB)-CCC

§RF244      AAA*BBB+CCC       ou     -AAA*BBB+CCC
§RF245      AAA*BBB-CCC       ou     -AAA*BBB-CCC
§RF246      AAA/BBB+CCC       ou     -AAA/BBB+CCC
§RF247      AAA/BBB-CCC       ou     -AAA/BBB-CCC

§RF250      AAA*(BBB+CCC)     ou     -AAA*(BBB+CCC)
§RF251      AAA*(BBB-CCC)     ou     -AAA*(BBB-CCC)
§RF252      AAA/(BBB+CCC)     ou     -AAA/(BBB+CCC)
§RF253      AAA/(BBB-CCC)     ou     -AAA/(BBB-CCC)

§RF260      AAA+(BBB*CCC)     ou     -AAA+(BBB*CCC)
§RF261      AAA-(BBB*CCC)     ou     -AAA-(BBB*CCC)
§RF262      AAA+(BBB/CCC)     ou     -AAA+(BBB/CCC)
§RF263      AAA-(BBB/CCC)     ou     -AAA-(BBB/CCC)

§RF264      AAA+BBB*CCC       ou     -AAA+BBB*CCC
§RF265      AAA-BBB*CCC       ou     -AAA-BBB*CCC
§RF266      AAA+BBB/CCC       ou     -AAA+BBB/CCC
§RF267      AAA-BBB/CCC       ou     -AAA-BBB/CCC

§RF270      (AAA+BBB)*CCC+DDD     ou     -(AAA+BBB)*CCC+DDD      (idem §RF230 avec DDD=0)
§RF271      (AAA-BBB)*CCC-DDD     ou     -(AAA-BBB)*CCC-DDD      (idem §RF231 avec DDD=0)
§RF272      (AAA+BBB)*CCC-DDD     ou     -(AAA+BBB)*CCC-DDD
§RF273      (AAA-BBB)*CCC+DDD     ou     -(AAA-BBB)*CCC+DDD

§RF280      (AAA+BBB)/CCC+DDD     ou     -(AAA+BBB)/CCC+DDD      (idem §RF232 avec DDD=0)
§RF281      (AAA-BBB)/CCC-DDD     ou     -(AAA-BBB)/CCC-DDD      (idem §RF233 avec DDD=0)
§RF282      (AAA+BBB)/CCC-DDD     ou     -(AAA+BBB)/CCC-DDD
§RF283      (AAA-BBB)/CCC+DDD     ou     -(AAA-BBB)/CCC+DDD
```

I : Divers Notations Et Autres Particularités du P30RS09

Instructions ANDCC et ORCC , possibilité de mettre directement, dans l'opérande, le nom des indicateurs du registre CC.
Exemple :

```
ORCC    I,F        ; permet de mettre les bits 6 et 4 du registre CC à 1
```

L'ordre des bits n'est pas important dans l'opérande, il faut simplement ne pas mettre deux fois le même indicateur, sous peine d'une erreur. Les bits du registre CC du 6809 sont : **E F H I N Z V C**

J : Exemple d'un format des lignes du programme source après assemblage

```
1        10        20        30        40        50        60        70        80        90        100       110
----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+----|----+----|---
LLLLL  AAAA cccc oooooo BBBB  EEEEEEEE MMMMMM OOOOOOOOOOOOOOOOOO CCCCCCCCCCCCCCC
|      |    |    |      |     |        |      |                  |
|      |    |    |      |     |        |      |                  CCCCCCCCCCCCCCC = Commentaire (Nb Car. Variable)
|      |    |    |      |     |        |      |
|      |    |    |      |     |        |      OOOOOOOOOOOOOOOOOO = Opérande (20 car)  (Opérande mini 10, maxi 70)
|      |    |    |      |     |        MMMMMM = Mnémonique (6 car)
|      |    |    |      |     |
|      |    |    |      |     EEEEEEEE = Etiquette (8 car)  (Etiquette mini 6, maxi 20)
|      |    |    |      |
|      |    |    |      BBBB = Adresse Branchement en Hexa (4 car)
|      |    |    |
|      |    |    oooooooo = Opérande en Hexa (7 car) ------------------> mis dans code Objet
|      |    |
|      |    cccc = Mnémonique en Hexa (4 car) -------------------------> mis dans code Objet
|      |
|      AAAA = Adresse en Hexa (4 car)
|
LLLLL = Numéro de Ligne (5 car)
```

A : Généralités

Le fichier sélectionné est lu octet par octet. Le décodage de ces informations en hexa, se fait en fonction de la table des mnémoniques du 6809.

Avant de lancer le désassemblage, il faut renseigner la zone **[Adresse d'Implantation]** par une valeur en Hexa sur 16 bits (4 chiffres) sans mette le signe $

Le résultat du désassemblage est dans un autre fichier, dans le même répertoire, avec le même nom mais complémenté du suffixe suivant : **[P30RS09 vX.XX desassemblage AAMMJJ_HHMM].txt**

AAMMJJ_HHMM date et heure du traitement
(AA = année, MM = mois, JJ = jour, HH = heure, MM = minute)

Les noms des étiquettes, après désassemblage, sont au format E_xxxx
Avec E_ ..... comme Etiquette et ....xxxx est la valeur en Hexa de l'adresse de l'étiquette.

A la fin du désassemblage, il est fort probable qu'il y ait des erreurs, car il est très difficile de déterminer les zones de data, générées par les directives d'assemblage FCB, FDB ou FCC.

B : Balises

Néanmoins, il est possible de réduire ce nombre d'erreurs en incorporant des balises dans le fichier hexa, afin d'encapsuler les chaînes de caractères. Un peu du même état d'esprit que la programmation en XTML.
Ces chaînes de caractères sont visibles par l'intermédiaire d'un éditeur Hexadécimal.

ATTENTION : La saisie des balises nécessite une grande rigueur. Les balises sont toujours sur deux caractères (fichier hexa oblige). Le format est le suivant :

Balises d'ouvertures		Balise. unique de fermeture
{	.....	**}}**
{-	balise pour la directive **FCB**	
{=	balise pour la directive **FDB**	Pour ces six balises
{_	balise pour la directive **FCC**	on trouvera à la fin
{;	balise mettant un Commentaire à partir de la zone Etiquette	la balise de fermeture
{*	balise mettant un Commentaire à partir de la zone Opérande	**}}**
{/	balise mettant un Commentaire à partir de la zone Commentaire	

Exemple

Dans le fichier hexa d'origine

```
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF204D6F6E6974657572205041B33202056312E3033206C6520313
92E362E383900FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

Dans le fichier d'origine vu par un éditeur hexa

```
00007F70  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00007F80  20 4D 6F 6E 69 74 65 75 72 20 50 41 4B 33 20 20    Moniteur PAK3
00007F90  56 31 2E 30 33 20 6C 65 20 31 39 2E 36 2E 38 39   V1.03 le 19.6.89
00007FA0  00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   .ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```

Pour faire une recherche dans le fichier hexa d'origine, par copier-collé, il faut supprimer les espaces provenant de l'éditeur hexa, pour cela ouvrir la fenêtre **Traitement sur Fichiers** et utiliser l'outil **H**

Mise en place des balises

```
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF{_204D6F6E6974657572205041 4B33202056312E3033206C65203
1392E362E3839}}{-00}}FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

Résultat du désassemblage

```
14583                           ORG     $FF80
14584   FF80                    FCC     " Moniteur PAK3  V1.03 le 19.6.89"
14585   FFA0                    FCB     $00
```

C : Zones non programmées

Lors du désassemblage on peut trouver des zones avec des octets "non programmés", ces octets sont mis à $FF.

Ces zones peuvent être au tout début ou à l'intérieur du fichier en hexa.

Si la zone est au début

Il y aura un commentaire dans le listing de désassemblage du type, suivi de d'une ligne EQU et d'une ligne ORG.

```
;#########################################################################
;#   Pour INFO, avant ce désassemblage il y a eu 2048 codes égal à $FF
;#                                                 (2048 = $0800)
;#       Adresse de début = $8000
;#       Adresse de fin   = $87FF
;#########################################################################

AdrDeb            EQU    8800          ; Ligne  EQU  servant de référence pour
                                       ; un futur assemblage par le P30RS09
                  ORG    AdrDeb        ;
```

ATTENTION : La ligne EQU est primordiale, en servant de repère cette ligne EQU va définir la position du début de la zone étiquette en vue d'un futur assemblage par le P30RS09.

Si la zone est à l'intérieur du listing

On aura un commentaire du type

```
FF0C 86   40              LDA    #$40                ;
                                                     ;
;#Block_FF#############################################################################
;#     Présence d'un Block d'octets à $FF, représentant 21 octets soit $0015
;#######################################################################################
                                                     ;
                          ORG    $FF23               ;
```

La dernière adresse est $FF0C + 2 = $FF0E et $FF0E + $0015 = $FF23 d'où le ORG $FF23

D : Tableau des vecteurs d'interruptions

Si le désassemblage arrive dans les adresses les plus hautes de l'espace adressable en 16 bits ($FFF0 à $FFFF), il y aura par exemple l'impression du tableau suivant :

```
FFEA 7E CB2A   E_FFEA           JMP  E_CB2A     ;
FFED 7D 7FE0                    FCC    "7D7FE0" ;<<<<<<< ; Code Hexa à désassembler
                                                ; manuellement, car à 3 octets
                                                ; avant la table des vecteurs !
                                                ; on pourrait mettre  TST  $7FE0
                                                ;  à la place de  FCC  "7D7FE0"
                                                ;
                                ORG  $FFF0      ;
FFF0    FFFF   Vecteur_Réservé  FDB  $FFFF      ; Vecteur d'interruption Réservé
FFF2    7D73   Vecteur_SWI3     FDB  $7D73      ; Vecteur d'interruption SWI3
FFF4    B9FE   Vecteur_SWI2     FDB  $B9FE      ; Vecteur d'interruption SWI2
FFF6    7D7C   Vecteur_FIRQ     FDB  $7D7C      ; Vecteur d'interruption FIRQ
FFF8    7D79   Vecteur_IRQ      FDB  $7D79      ; Vecteur d'interruption IRQ
FFFA    7D70   Vecteur_SWI      FDB  $7D70      ; Vecteur d'interruption SWI
FFFC    7D76   Vecteur_NMI      FDB  $7D76      ; Vecteur d'interruption NMI
FFFE    CA08   Vecteur_RESET    FDB  $CA08      ; Vecteur d'interruption RESET
```

E : Options de la fenêtre Désassemblage

Option  SANS (ou avec) les valeurs des registres dans les commentaires 
On met dans le champ commentaire, la valeur des registres.

Pour les adressages Immédiat, Direct, Etendu : Seul les registres A, B, D, X ou Y sont concernés.

Pour l'adressage Indexé : Seul les registres X et Y sont concernés et pour les instructions LEAX et LEAY.

Les Post-Bytes concerné sont les suivants :

Groupe	Post-Byte	Syntaxe	
03A	$88	n,X	8 bits
03A	$A0	n,Y	8 bits
04A	$89	n,X	16 bits
04A	$A9	n,Y	16 bits
08A	$80	,X+	
08A	$A0	,Y+	
09A	$81	,X++	
09A	$A1	,Y++	
10A	$82	,-X	
10A	$A2	,-Y	
11A	$83	,--X	
11A	$A3	,--Y	
12A	$8C	n,PCR	8 bits
12A	$AC	n,PCR	8 bits
12A	$CC	n,PCR	8 bits
12A	$EC	n,PCR	8 bits
13A	$8D	n,PCR	16 bits
13A	$AD	n,PCR	16 bits
13A	$CD	n,PCR	16 bits
13A	$ED	n,PCR	16 bits

Groupe	Post-Byte	Syntaxe
09B	$91	[,X++]
09B	$B1	[,Y++]

Groupe	Post-Byte	Syntaxe
11B	$93	[,--X]
11B	$B3	[,--Y]

Quelques exemples de syntaxes trouvées dans le champ commentaire.

Exemples 01

```
[A=$20-->$0A]   ===> [D=$20F0-->$0AF0]
```

Ce qui veut dire que l'accumulateur A passe de la valeur $20 à la valeur $0A
Entraînant automatiquement le passage de l'accumulateur D de la valeur $20F0 à $0AF0

Exemples 02

Lors d'un transfert de A dans DP

```
TFR  A,DP   ;  [A=$00] [DP=$EF-->$00]
```

L'accumulateur est inchangé, il a la valeur $00
Le registre DP passe de la valeur $EF à $00

Exemples 03

Pour une multiplication de l'accumulateur A avec l'accumulateur B

```
MUL         ;  [D=$4F10-->$04F0] ===> [A=$4F-->$04] [B=$10-->$F0]
```

L'accumulateur D valait $4F10 et passe à $04F0
Entraînant automatiquement le passage des l'accumulateurs :
A de la valeur $4F à $04
B de la valeur $10 à $F0

Option B AVEC (ou sans) les tableaux des symboles en fin de listing. 

Si cette option est cochée, on trouvera en fin de fichier, deux listes :
La première c'est la liste de toutes les adresses de branchement décelées dans le listing.

```
|---------------------liste des adresses de branchement------------
|F014  F020  F034  F000  F081  F366  F0B4  F95C  F0A7  F37A
|F375  F128  F10A  F132  F12A  F13A  F305  F176  F189  F180
|F7C6  F206  F1CC
```

La seconde on trouvera pour chaque étiquette de branchement, la ou les lignes faisant appel à cette étiquette.

```
|-----Numéro des lignes de programme pour les adresses de branchement---
|  Etiquettes                  N° des Lignes
|  ----------                  -------------
|E0_Branch_$F014  :  00297
|E0_Branch_$F081  :  00340
|E0_Branch_$F0B4  :  00352 00356
|E2_SProg__$F95C  :  00353 01285 01299
```

Option C AVEC (ou sans) impression de l'ensemble des erreurs: 

Si cette option est cochée, il y aura un récapitulatif de toutes les erreurs rencontrées lors du désassemblage.
Par exemple

```
|___________Désassemblage AVEC ERREUR(S)____avec 77 erreur(s)________
|  N°    |  N°     |
|Ligne   | Erreur  |          Libellé d'erreur
|        |         |
|00039   {S0401 : Le code $02 n'est pas dans la table des Mnémoniques !
|00040   {S0401 : Le code $CD n'est pas dans la table des Mnémoniques !
```

Option D AVEC (ou sans) impression du format d'une ligne. 

Si cette option est cochée, juste après le listing de désassemblage on trouvera le format d'une ligne.

F : Erreurs rencontrés lors d'un Désassemblage

Lors du désassemblage d'un fichier de code en hexa, on se heurte au problème des données insérées dans le code source.

C'est les directives FCB, FDB et FCC qui posent problème, en effet dans le code objet aucun artifice ne permet d'indiquer la présence de DATA.

Exemple d'un Assemblage pour obtenir un code objet et dans la foulé un désassemblage de ce code objet pour essayer d'obtenir le même listing source.

Code source d'origine

```
00917  F2CE 8D   B3      F283  ZOTCH2       BSR    SEND
00918  F2D0 0C   90            ZOTCH3       INC    <SWICNT
00919  F2D2 3B                              RTI
00920
00933  F2D3 04                 ZPCRLS       FCB    $04
00934
00935  F2D4 30   8C FC         ZPCRLF       LEAX   <ZPCRLS,PCR
00951  F2D7 86   0D            ZPDATA       LDA    #CR
00952  F2D9 8D   A8      F283               BSR    SEND
00953  F2DB 86   0A                         LDA    #LF
00972  F2DD 8D   A4      F283  ZPDTLP       BSR    SEND
00973  F2DF A6   80            ZPDTA1       LDA    ,X+
00974  F2E1 81   04                         CMPA   #EOT
00975  F2E3 26   F8      F2DD               BNE    ZPDTLP
00995  F2E5 8D   28       F30F  ZPAUSE      BSR    XQPAUS
00996  F2E7 8D   06      F2EF               BSR    CHKABT
00997  F2E9 1F   A9                         TFR    CC,B
00998  F2EB E7   E4                         STB    0,S
00999  F2ED 20   E1      F2D0               BRA    ZOTCH3
```

Pas du tout le même code source

Code objet suite à l'assemblage du listing ci-dessus

```
8DB30C903B04308CFC860D8DA8860A8DA4A680810426F88D288D061FA9E7E420E1
```

Listing suite au désassemblage du code objet ci-dessus

```
00606  F2CE 8D   B3      F283  E0_Branch_$F2CE  BSR    E1_SProg_$F283
00607  F2D0 0C   90            E0_Branch_$F2D0  INC    <Const_EF90
00608  F2D2 3B                                  RTI
00609
00610
00611                                 ;==============================
00612  F2D3 04   30                               LSR    <Const_EF30
00613  F2D5 8C   FC86                             CMPX   #Const_FC86
00614  F2D8 0D   8D                               TST    <Const_EF8D
00615  F2DA A8   86                               EORA   A,X
00616  F2DC 0A   8D                               DEC    <Const_EF8D
00617  F2DE A4   A6                               ANDA   A,Y
00618  F2E0 80   81                               SUBA   #$81
00619  F2E2 04   26                               LSR    <Const_EF26
00620  F2E4 F8   8D28                             EORB   >Const_8D28
00621  F2E7 8D   06      F2EF               BSR    E1_SProg_$F2EF
00622  F2E9 1F   A9                         TFR    CC,B
00623  F2EB E7   E4                         STB    0,S
00624  F2ED 20   E1      F2D0               BRA    E0_Branch_$F2D0
```

ATTENTION : Les lignes commençant par le caractère | ($7C) ne sont pas prises en compte

A : TRAITEMENT A Ajout de numéros de ligne sur 5 colonnes à gauche de chaque ligne

Pour chaque ligne du fichier sélectionné, sauf celles qui débutent par le caractère | ($7C), on ajoute un numéro de ligne sur 5 caractères.
Le résultat du traitement est dans un autre fichier, dans le même répertoire, avec le même nom mais complémenté du suffixe suivant : **[P30RS09 vX.XX Trait-A AAMMJJ_HHMM].txt**
(AA = année, MM = mois, JJ = jour, HH = heure, MM = minute)

B : TRAITEMENT B Suppression de N colonnes à gauche pour chaque ligne

Pour chaque ligne du fichier sélectionné, sauf celles qui débutent par le caractère | ($7C), on supprime un nombre **N** caractères à partir de la gauche. **N** étant renseigné par l'opérateur.
Le résultat du traitement est dans un autre fichier, dans le même répertoire, avec le même nom mais complémenté du suffixe suivant : **P30RS09 vX.XX Trait-B AAMMJJ_HHMM].txt**
(AA = année, MM = mois, JJ = jour, HH = heure, MM = minute)

C : TRAITEMENT C Insertion de N caractères SPACE à une position déterminé pour chaque ligne

Pour chaque ligne du fichier sélectionné, on insère un nombre **N** de caractères **SPACE** entre les colonnes renseignées par l'opérateur. **N** est également renseigné par l'opérateur.
Le résultat du traitement est dans un autre fichier, dans le même répertoire, avec le même nom mais complémenté du suffixe suivant : **P30RS09 vX.XX Trait-C AAMMJJ_HHMM].txt**
(AA = année, MM = mois, JJ = jour, HH = heure, MM = minute)

D : TRAITEMENT D Insertion d'une Tabulation à une position déterminée pour chaque ligne

Pour chaque ligne du fichier sélectionné, on insère un caractère **TABULATION** ($09) entre les colonnes renseignées par l'opérateur.
Le résultat du traitement est dans un autre fichier, dans le même répertoire, avec le même nom mais complémenté du suffixe suivant : **P30RS09 vX.XX Trait-D AAMMJJ_HHMM].txt**
(AA = année, MM = mois, JJ = jour, HH = heure, MM = minute)

E : TRAITEMENT E Suppression des caractères SPACE dans chaque ligne

Dans tout le fichier sélectionné, on supprime tous les caractères **SPACE** ($20).
Le résultat du traitement est dans un autre fichier, dans le même répertoire, avec le même nom mais complémenté du suffixe suivant : **P30RS09 vX.XX Trait-E AAMMJJ_HHMM].txt**
(AA = année, MM = mois, JJ = jour, HH = heure, MM = minute)

F : TRAITEMENT F Remplacement des TABULATIONS par des caractères SPACES dans chaque ligne

Dans tout le fichier sélectionné, on remplace tous les caractères **TABULATION** ($09) par des caractères **SPACE** ($20).
Le résultat du traitement est dans un autre fichier, dans le même répertoire, avec le même nom mais complémenté du suffixe suivant : **P30RS09 vX.XX Trait-F AAMMJJ_HHMM].txt**
(AA = année, MM = mois, JJ = jour, HH = heure, MM = minute)

G : TRAITEMENT G Conversion Fichier BINAIRE en Fichier en HEXADECIMAL

A partir du fichier sélectionné, il y a conversion de tous les caractères par leurs équivalences en hexadécimal.
Exemple : `¼‹H_Ž~ o€Z` converti en `BD8B485F8E7E006F805A`
Le résultat du traitement est dans un autre fichier, dans le même répertoire, avec le même nom mais complémenté du suffixe suivant : **P30RS09 vX.XX Trait-G AAMMJJ_HHMM].txt**
(AA = année, MM = mois, JJ = jour, HH = heure, MM = minute)

H : TRAITEMENT H Détection des lignes SANS points-virgules (pour les commentaires)

Dans tout le fichier sélectionné, on recherche les lignes n'ayant pas de marqueur signalant début du champ commentaire. Ce sont les lignes n'ayant de point-virgule.
Seul ces lignes sont mises dans le fichier résultat.
Le résultat du traitement est dans un autre fichier, dans le même répertoire, avec le même nom mais complémenté du suffixe suivant : **P30RS09 vX.XX Trait-H AAMMJJ_HHMM].txt**
(AA = année, MM = mois, JJ = jour, HH = heure, MM = minute)

I : TRAITEMENT I Suppression des colonnes, entre la colonne X et la colonne Y inclus

Pour chaque ligne du fichier sélectionné, les caractères inclus entre la position X et la position Y sont supprimés.
X et Y ne pouvant pas être supérieur à la longueur de la plus grande ligne du fichier sélectionné.
X ne peut pas être supérieur à Y et X ne peut pas être égal à Y.
Le résultat du traitement est dans un autre fichier, dans le même répertoire, avec le même nom mais complémenté du suffixe suivant : **P30RS09 vX.XX Trait-I AAMMJJ_HHMM].txt**
(AA = année, MM = mois, JJ = jour, HH = heure, MM = minute)

J : TRAITEMENT J Suppression des SPACES dans une seule ligne

Suppression des caractères ESPACES d'une chaîne de caractère mise dans le champ **[Ligne à Traiter]**.

Pour information :

Dès que le copier-coller est réalisé, le traitement est automatiquement affiché.

Dés que l'on clique sur le champ **[Ligne à traiter]**, il y a effacement de ce champ, permettant un nouveau copier-coller.

Exemple :

Un copier-coller venant d'un éditeur en HEXA . . . . . `45 72 72 65 75 72 20 64 65`

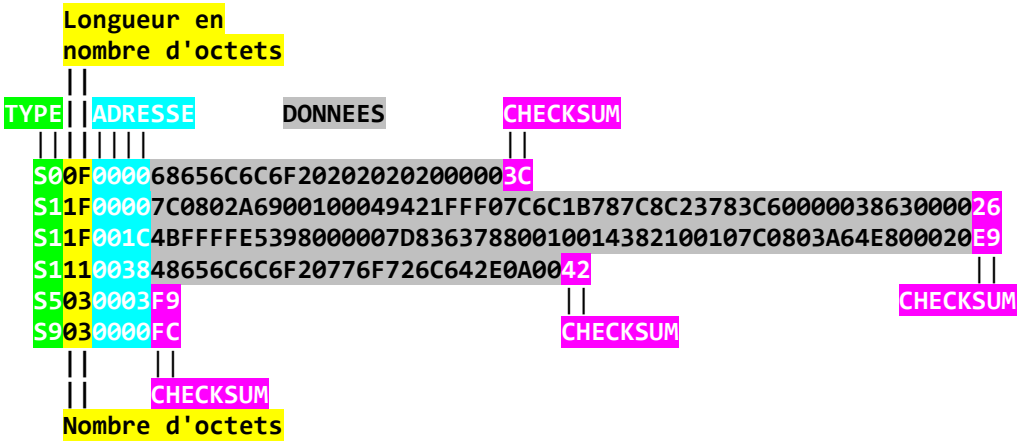
Dans le champ **[Conversion en ASCII]** :
On trouvera la conversion en caractères ASCII . . . . . `Erreur de`

Dans le champ **[Après Traitement]** :
On trouvera la ligne sans espaces . . . . . `457272657572206465`

Ce qui permet de refaire un copier-coller à partir du champ **[Après Traitement]** pour en faire une recherche dans un fichier en hexa.

S1S9 également connu sous le nom S-Record ou SREC, est un format de représentation de fichier binaire en ASCII.
Développé dans les années 1970 par la société MOTOROLA. Il était utilisé alors pour la programmation du microprocesseur MOTOROLA 6800.
Le format textuel offre de nombreux avantages sur le format binaire : il peut être imprimé, inspecté ou modifié avec un éditeur de texte ordinaire.
Ces fichiers sont utilisés pour le transfert de programmes vers les programmateurs d'EPROM en communication série (RS-232). Il est toujours utilisé en informatique embarquée ainsi que son équivalent, le format HEX développé par la société Intel.

La structure de chaque ligne est la suivante :



Champ TYPE (sur 2 caractères)

Exemple : S1130000285F245F2212226A000424290008237C2A

S Le caractère débutant l'enregistrement (S comme "Short")
X Un chiffre de 0 à 9, définissant le type d'enregistrement.

Il existe 8 types d'enregistrements :

Type	Description	Octets d'adresse	Séquence de données
S0	En-tête	2	Oui
S1	Enregistrement avec zones adresses de 16 bits	2	Oui
S2	Enregistrement avec zones adresses de 24 bits	3	Oui
S3	Enregistrement avec zones adresses de 32 bits	4	Oui
S5	Nombre d'enregistrements dans le champ adresse	2	Non
S6	Nombre d'enregistrements dans le champ adresse	3	Non
S7	Fin de fichier pour un enregistrement S3	4	Non
S8	Fin de fichier pour un enregistrement S2	3	Non
S9	Fin de fichier pour un enregistrement S1	2	Non

S0

Chaîne d'en tête
Contient des données spécifiques au fournisseur tel que le nom du programme ou le numéro de version plutôt que des données binaire utile au programme.
Le champ Adresse est normalement à 0.

S1, S2, S3 Séquence de données, en fonction de la taille de l'adresse nécessaire.

- **S1** Pour un système de 16 bits à 64 Ko utilise S1 (et S9 pour la fin du bloc).
  La zone adresse est sur 2 octets exemple F35C

- **S2** Pour un système de 24 bits à 16 Mo utilise S2 (et S8 pour la fin du bloc).
  La zone adresse est sur 3 octets exemple 1BFF3D

- **S3** Pour un système de 32 bits à 4 Go utilise S3 (et S7 pour la fin du bloc).
  La zone adresse est sur 4 octets exemple 143ABB2F

S5

Indique sur le nombre de chaînes S1, S2 ou S3 transmises dans un bloc.
Le nombre d'enregistrements est mémorisé dans le champ d'adresse de 2 octets.
Il n'existe pas de champs de données associés à ce type d'enregistrement.

S6

Indique sur le nombre de chaînes S1, S2 ou S3 transmises dans un bloc.
Le nombre d'enregistrements est mémorisé dans le champ d'adresse de 3 octets.
Il n'existe pas de champs de données associés à ce type d'enregistrement.

S7, S8, S9 Le champ d'adresse des dossiers, où peut contenir une adresse de départ pour le programme.

- **S7** Chaîne finale d'un bloc de chaînes S3
  L'adresse peut optionnellement contenir une adresse sur 4 octets qui pointe sur une instruction.
  Il n'y a aucune donnée.

- **S8** Chaîne finale d'un bloc de chaînes S2
  L'adresse peut optionnellement contenir une adresse sur 3 octets qui pointe sur une instruction.
  Il n'y a aucune donnée.

- **S9** Chaîne finale d'un bloc de chaînes S1
  L'adresse peut optionnellement contenir une adresse sur 4 octets qui pointe sur une instruction.
  Il n'y a aucune donnée
  Si cette valeur n'est pas précisée, on devra utiliser la première entrée dans le code objet.

Champ LONGUEUR (sur 2 caractères)

Exemple : S113 0000 285F245F2212226A000424290008237C 2A

Nombre de paires de caractères de la chaîne, exprimé en Hexa
On ne tient pas compte du champ TYPE et du champ LONGUEUR
On commence à compter à partir du champ ADRESSE en incluant le champ DONNEE et CHECKSUM

Champ ADRESSE (sur 4, 6 ou 8 caractères)

Exemple : S113 0000 285F245F2212226A000424290008237C 2A

C'est l'adresse mémoire où doivent êtres rangées les données.
Le premier octet de données de la ligne est stocké dans l'adresse indiquée ici par 0000.

Après le stockage de cet octet de donnée l'adresse est ensuite incrémentée pour l'octet suivant et ainsi de suite, jusqu'à la fin de la ligne.

La longueur de la zone adresse dépend du type d'Enregistrement.
- Pour le type S1 la zone adresse est de 2 octets exemple F35C

- Pour le type S2 la zone adresse est de 3 octets exemple 1BFF3D
- Pour le type S3 la zone adresse est de 4 octets exemple 143ABB2F

L'adresse est envoyée avec le MSB d'abord.
Le fichier peut également contenir des sauts d'adresse, pour une partie de mémoire inutilisée.

Champ DONNEES (sur 0 à n caractères)

Exemple : S1130000**285F245F2212226A000424290008237C**2A

De 0 a n octets. Séquence de paires de chiffres hexadécimaux représentant les octets de données.
Une règle consiste à limiter la quantité de données par chaîne à 28 octets donc à 56 caractères

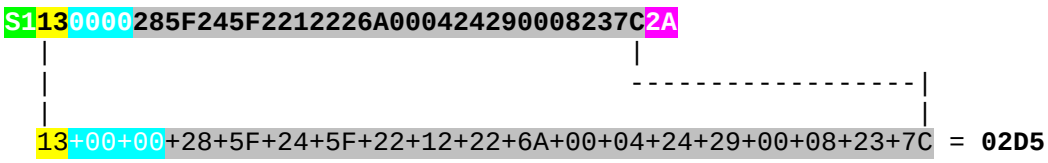
Champ CHECKSUM (sur 2 caractères)

Exemple : S1130000285F245F2212226A000424290008237C**2A**

Somme de contrôle (en anglais, CheckSum), deux chiffres hexadécimaux, octet de vérification ou somme de contrôle.

Pour le calcul du CheckSum :

1) Faire la somme de tous les octets des champs **LONGUEUR**, **ADRESSE**, **DONNEES**.



2) On trouve la valeur **$02D5**, à partir de cette valeur on prend l'octet le moins significatif (LSB Least Signifiant Bit), c'est à dire **$D5**

3) Faire le complément à 1 de **$D5** l'octet le moins significatif (LSB)

```
                          $D5 = % 1101 0101
Complément à 1 de $D5 = % 0010 1010 = $2A
```

4) Ce qui donne le champ Checksum = **$2A**