CPLECAM

Système de développement

Pour LECAM

Version 1
1991

Manuel Utilisateur

SOMMAIRE

PRESENTATION

CPLECAM est un environnement de développement permettant de réaliser et de mettre au point des programmes destinés au **LECAM**, le lecteur de carte à mémoire de France-Télécom.

Cet environnement de développement comporte un éditeur de texte pleine page, un compilateur intégré et un débogueur symbolique.

L'EDITEUR DE TEXTE

L'éditeur de texte permet la saisie de textes ASCII et comporte les fonctions de traitement les plus couramment proposées par les standards du marché. Ses principales caractéristiques sont :

-la taille des fichiers à éditer n'est limitée que par la place disponible en mémoire de travail du micro-ordinateur,

-il permet de travailler sur deux fichiers simultanément,

-il dispose des fonctions de recherche, remplacement, manipulation de blocs, déplacements par caractères, mots, pages, etc...

-il accepte les écrans de 80 à 132 colonnes avec un nombre de lignes quelconque.

LE COMPILATEUR

Le compilateur traduit le programme source pour LECAM en son équivalent exécutable par le LECAM. Il dispose de pseudos instructions facilitant l'écriture et la maintenance des programmes développés. En cas d'erreur lors de la compilation, la main est rendue à l'éditeur et le curseur est positionné à l'endroit où elle a été détectée.

Lors de la compilation, tous les fichiers nécessaires au débogueur symboliques ainsi qu'aux programmes d'applications qui utilisent les programmes LECAM sont générés. La structure de ces fichiers est fourni en annexe.

LE DEBOGUEUR

Le débogueur permet de dialoguer avec un LECAM, de télécharger un ou plusieurs programmes, d'en demander l'exécution, de poser des points d'arrêts, etc... et plus généralement, d'émettre des consignes vers le LECAM et d'en recevoir les réponses.

STRUCTURE DU DOCUMENT

L'organisation de ce manuel reprend par chapitre, le découpage par paragraphe adopté dans cette présentation. La sémantique du langage LECAM, ses spécifications, ses particularités de fonctionnement ne font pas partie de ce document et sont à se procurer auprès du CNET[1]. Par contre, la syntaxe du langage telle qu'elle est acceptée par le compilateur est décrite dans le chapitre consacré au compilateur.

1S.T.U.C.A.M Spécifications Techniques d'Utilisation du LECAM. CNET PARIS A, Département MGA/DCT, Pièce 331-A, 38, 40 rue du Général Leclerc, 92131 ISSY les MOULINEAUX. (150 FHT en 1990).

INSTALLATION ET LANCEMENT

INSTALLATION

Placer la disquette n° 1 dans un des lecteurs de disquette de votre micro-ordinateur et sélectionner ce lecteur (**A:** ou **B:** par exemple). Tapez **INSTALL** et répondez aux questions qui vous sont posées :

-Nom du disque destinataire (**C:** par défaut),

-Nom du répertoire destinataire (**\CPLECAM** par défaut),

LANCEMENT

Une fois l'installation terminée, positionnez vous sur le disque et le répertoire sur lequel se trouve le système de développement et entrez **CPLECAM** pour exécuter le programme.

Vous pouvez également spécifier au lancement le nom d'un programme devant être édité en tapant **CPLECAM <nom du programme>**.

La figure ci-dessous représente le contenu de l'écran lorsque le programme est lancé.

inserer figure

UTILISATION DE L'EDITEUR DE TEXTE

GENERALITES

L'éditeur de texte est le pivot du système de développement. C'est lui qui est exécuté au lancement du système. Il permet de créer et modifier des fichiers de textes ainsi que de lancer le compilateur et le débogueur.

L'éditeur permet de travailler simultanément sur deux fichiers situés dans deux fenêtres. Au lancement du programme, seule la première fenêtre est présente sur l'écran. Il est possible de passer globalement d'une fenêtre à l'autre (voir **Fenêtre** dans le présent chapitre). Dans ce cas, toutes les lignes disponibles à l'écran sont utilisées pour visualiser le contenu de la fenêtre active. Il est également possible d'afficher une portion de chacune des fenêtre à l'écran et d'en modifier la taille.

L'éditeur supporte des lignes de 132 colonnes maximum. Selon l'écran utilisé, tout ou partie de ces colonnes sont affichées. Dans tous les cas, il est impératif d'utiliser un moniteur et un contrôleur capable d'afficher au moins 80 colonnes.
Les touches de déplacement de curseur permettent de visualiser les colonnes non affichées si le moniteur ne permet pas l'affichage en 132 colonnes.

L'éditeur fait un large usage des menus déroulants pour faciliter l'accès à certaines commandes. Lorsqu'un tel menu est proposé, chaque ligne du menu correspond à une option possible. Pour sélectionner l'une d'entre elle, il suffit de positionner le curseur sur l'option désirée et d'appuyer sur **[RC]**. Le déplacement du curseur s'effectue par l'appui sur les touches **[↑], [↓], [→], [←]**. Il est également possible de saisir directement le caractère marqué en rouge (ou en surbrillance sur un moniteur monochrome) pour l'option désirée.

De manière générale, il est possible d'arrêter une fonction en cours en appuyant sur la touche **[Esc]**.

ECRAN DE L'EDITEUR

L'éditeur se présente de la façon suivante :

-la première ligne de l'écran est un bandeau rappelant les principales touches de fonctions actives. Elles sont associées à un mnémonique correspondant aux principales fonctions proposées par l'éditeur,

-la deuxième ligne fournit des indications dynamiques correspondant à l'état courant de l'éditeur :

♦**Lig** : n° de ligne courant où se trouve le curseur,

♦**Col** : n° de colonne courant où se trouve le curseur,

♦**Insert** : si cet indicateur est présent, il signifie que l'éditeur se trouve en mode insertion. Tous les caractères saisis s'insèrent dans le texte déjà tapé. S'il est absent, l'éditeur se trouve en mode écrasement. Les caractères saisis remplacent ceux qui se trouvent à la position du curseur au moment de la saisie.

♦**Indent** : si cet indicateur est présent, l'éditeur se trouve en mode indentation automatique. L'insertion d'une nouvelle ligne positionne le curseur sous le premier caractère différent d'un espace de la ligne précédente. S'il est absent, l'insertion d'une nouvelle ligne positionne le curseur en colonne 1.

♦**Editeur** : cet indicateur précise que l'on se trouve sous l'éditeur de texte du système de développement. Son autre valeur possible est **débogueur**.

♦**<nom de fichier>** : il s'agit du nom du fichier de la fenêtre active. Si aucun fichier n'est chargé, la valeur de <nom de fichier> est **NONAME**.

C'est également cette ligne qui est utilisée lors de l'affichage de messages d'erreurs ou d'informations.

-les autres lignes sont destinées à afficher le ou les fichiers en cours d'édition.

AIDE - F1

AIDE EN LIGNE

L'appui sur la touche **[F1]** permet de disposer d'une aide en ligne sur les fonctions proposées par l'éditeur.

-lorsqu'un message d'erreur est affiché, l'aide fourni une explication ou des commentaires sur ce message,

-lorsqu'aucun message d'erreur est affiché, le premier appui sur la touche **[F1]** fournit une aide sur les principales commandes de l'éditeur et du débogueur. Un deuxième appui sur la touche **[F1]** provoque

l'apparition d'un sommaire. Les touches de déplacement du curseur permettent de sélectionner l'item à propos duquel une aide est souhaitée.

-lorsqu'un mot apparait en jaune (ou en surbrillance sur un écran monochrome) il est possible de le sélectionner (positionnement du curseur sur le mot et appui sur la touche **[RC]**) afin d'obtenir une information complémentaire concernant cet item.

FICHIERS - F3

FICHIERS

L'appui sur la touche **[F3]** provoque l'affichage d'un menu déroulant permettant d'accéder aux fonctions suivantes :

-**Chargement d'un fichier** : permet l'accès à la fonction de chargement d'un fichier. Une fenêtre est affichée permettant de saisir le nom du fichier à charger. Plusieurs possibilités sont autorisées :

-il est possible de saisir le nom complet du fichier. S'il existe, un message d'information "Fichier en cours de chargement" est affiché. Lorsque le fichier est chargé, l'indicateur <nom de fichier> de la ligne d'information prend la valeur du nom du fichier.

-il est possible d'utiliser les méta-caractères proposés par MS-DOS. Ainsi, ***.LEC** demande à l'éditeur le chargement d'un fichier dont le suffixe est **LEC**. Si plusieurs fichiers répondent à ce critère, l'éditeur affiche les noms correspondants à l'intérieur d'une fenêtre. Pour sélectionner le fichier à charger, il faut positionner le curseur sur le nom du fichier voulu et appuyer sur **[RC]**. Les touches **[↑], [↓], [→], [←]** permettent de déplacer le curseur, nom de fichier par nom de fichier. Les touches **[▲]** et **[▼]** permettent de déplacer le curseur page par page à l'intérieur de la fenêtre. La touche **[Fin]** positionne le curseur sur le dernier nom de fichier répondant aux critères. La touche **[Home]** positionne le curseur sur le premier nom de fichier répondant aux critères.

Limitation : 100 fichiers maximum peuvent être sélectionnés selon ce principe.

Si un fichier est déjà chargé au moment ou l'on active cette fonction, l'éditeur propose qu'il soit sauvegardé (à condition qu'il ait été modifié).

-**nouveau** : permet de créer un nouveau fichier dans la fenêtre active. Si un fichier est déjà chargé au moment où l'on active cette fonction, l'éditeur en propose qu'il soit sauvegardé (à condition qu'il ait été

modifié).

-**Sauve** : identique à la fonction "sauve" accessible par la touche **[F2]**.

-**Quit** : identique à la fonction de sortie accessible par la combinaison de touches **[Alt][X]**.

SAUVEGARDE - F2

SAUVEGARDE D'UN FICHIER

L'appui sur la touche **[F2]** permet de sauvegarder sur fichier le contenu de la fenêtre active. Une fenêtre est affiché et un nom de fichier est proposé. Il est possible de modifier ce nom. L'appui sur la touche **[RC]** sauvegarde le fichier sous le nom indiqué dans la fenêtre. L'appuie sur la touche **[Esc]** permet d'abandonner la sauvegarde et de retourner sous l'éditeur.

OPTIONS - F4

OPTIONS

L'appui sur la touche **[F4]** affiche un menu déroulant et permet d'accéder à certaines options de configuration de l'éditeur :

-**Tabulation** : il est possible de préciser un nombre d'espace qui sera inséré lors de l'appui sur la touche **tabulation**. Les valeurs possibles vont de 0 à 99.

-**Indentation** : permet d'activer ou de désactiver l'indentation automatique.

-**Rffu** : réservé.

Ces options sont sauvegardées lorsque l'on sort du système de développement.

FENETRES - F7

FENETRES

L'appui sur la touche **[F7]** affiche un menu déroulant permettant de sélectionner la fenêtre de travail.

-**Fenêtre** : la sélection de cet option permet de changer la fenêtre active.

-**Règle** : la sélection de cette option permet de séparer l'écran en deux fenêtres. Une barre constituée des caractères ▾▾▾...▾▾▾ (fenêtre basse active) ou ▴▴▴...▴▴▴ (fenêtre haute active) est affichée au milieu de l'écran. Les touches **[**↑**]** et **[**↓**]** permettent de la déplacer afin de disposer de tailles de fenêtres correspondant au mieux aux besoins.

-**Effacer règle** : la sélection de cette option permet de supprimer la règle. Tout l'écran est alors disponible pour l'affichage de la fenêtre active.

BLOCS - F6

BLOC

L'appui sur la touche **[F6]** affiche un menu déroulant permettant d'accéder aux fonctions de manipulation de blocs.

-**marquage** : la sélection de cette option permet de commencer le marquage d'un bloc. Le début du marquage se fait à la position courante du curseur. Pour marquer un bloc, il suffit de déplacer le curseur jusqu'au dernier caractère appartenant à ce bloc.

-**Annule marquage** : la sélection de cette option permet d'annuler le marquage en cours.

-**Efface** : la sélection de cette option permet d'effacer le bloc en marqué.

-**Déplace** : la sélection de cette option copie le bloc marqué dans un buffer temporaire et l'efface de la fenêtre d'édition.

-**Copie** : la sélection de cette option copie le bloc marqué dans un buffer temporaire.

-**Retrouve** : la sélection de cette option recopie le buffer temporaire à la position du curseur.

-**Disque** : la sélection de cette option copie le bloc marqué sur disque. L'utilisateur doit préciser le nom du fichier qui recevra ce bloc.

COMPILATION - F9

COMPILATION

L'appui sur la touche **[F9]** lance la compilation du fichier chargé dans la fenêtre active. Une fenêtre est affiché au milieu de l'écran et précise :

-le niveau du passage effectué par le compilateur (PASSE 1 ou PASSE 2),

-le numéro de ligne en cours de compilation.

La compilation peut être arrêtée par l'appui sur la touche **[Esc]**.

DEBOGUEUR - F8

DEBOGUEUR

L'appui sur la touche **[F8]** lance le débogueur symbolique (pour plus de précision, voir le chapitre consacré au débogueur).

EFFACEMENT DE LIGNE - [Ctrl][Y]

EFFACEMENT D'UNE LIGNE

L'appui sur les touches **[Ctrl] [Y]** efface la ligne sur laquelle est positionné le curseur.

RECHERCHE - [Ctrl][Q][F]

RECHERCHE

l'appui sur les touches **[Ctrl][Q]** et **[F]** permettent d'accéder à la fonction de recherche de l'éditeur.
A l'invite **Cherche :**, il faut saisir le texte à rechercher.
A l'invite **Option :**, il faut indiquer les options de recherches. Toutes les options sont facultatives.
Lorsque plusieurs options sont présentes, elles doivent être séparées par le caractère **Espace**.
La recherche s'effectue toujours "en avant" (du numéro de ligne le moins élevé vers le plus élevé, du numéro de colonne le moins élevé vers le plus élevé).

-**g** : recherche le texte à partir de la première ligne du fichier. Si cette option est absente, la recherche démarre à partir de la position courante du curseur.

-**co=<colonne 1>, <colonne 2>** : si cette option est présente, elle permet de préciser un intervalle de colonnes à l'intérieur desquelles s'effectuera la recherche. Si **<colonne 1>** est absent, la valeur par défaut correspondra à la colonne 1.
Si **<colonne 2>** est absent, la valeur par défaut correspondra à la colonne 132.
Si **<colonne 1>** ou **<colonne 2>** sont égales au caractère **.**, la valeur correspondra à la position courante du curseur.

-**lg=<ligne 1>, <ligne 2>** : cette option est équivalente à la précédente pour les numéros de lignes.

Les options **lg** et **co** sont prioritaires sur l'option **g**.

Exemples :co=10,. : colonnes 10 à 132
co=10,20 lg=30,40: colonnes 10 à 20, lignes 30 à 40.

REMPLACEMENT - [Ctrl][Q][A]

REMPLACEMENT

l'appui sur les touches **[Ctrl][Q]** et **[A]** permettent d'accéder à la fonction de remplacement de l'éditeur.

A l'invite **Cherche :**, il faut saisir le texte à rechercher.

A l'invite **Remplace :**, il faut saisir le texte qui remplacera le texte à rechercher.

A l'invite **Option :**, il faut indiquer les options de recherches. Toutes les options sont facultatives.

Lorsque plusieurs options sont présentes, elles doivent être séparées par le caractère **Espace**.

La recherche s'effectue toujours "en avant" (du numéro de ligne le moins élevé vers le plus élevé, du numéro de colonne le moins élevé vers le plus élevé).

-**g** : lorsque cette option est présente, la recherche débute à partir de la première ligne du fichier et se termine à la fin du fichier. Toutes les occurrences du texte à chercher sont susceptibles d'être remplacées. Si cette option est absente, la recherche démarre à la position courante du curseur et s'arrête à la première occurrence du texte cherché. Seul ce texte sera susceptible d'être remplacé.

-**c** : lorsque cette option est présente, une demande de confirmation sera proposée avant d'effectuer le remplacement.

-**co=<colonne 1>, <colonne 2>** : si cette option est présente, elle permet de préciser un intervalle de colonnes à l'intérieur desquelles s'effectuera la recherche. Si **<colonne 1>** est absent, la valeur par défaut correspondra à la colonne 1.

Si **<colonne 2>** est absent, la valeur par défaut correspondra à la colonne 132.

Si **<colonne 1>** ou **<colonne 2>** sont égales au caractère **.**, la valeur correspondra à la position courante du curseur.

-**lg=<ligne 1>, <ligne 2>** : cette option est équivalente à la précédente pour les numéros de lignes.

Les options **lg** et **co** sont prioritaires sur l'option **g**.

INSERTION/ECRASEMENT - [Ins]

INSERTION ACTIVE/INACTIVE

L'appui sur la touche **[Ins]** rend le mode insertion actif s'il était inactif et inactif (écrasement) s'il était actif.

SUPPRESSION - [Suppr]

SUPPRESSION CARACTERE

L'appui sur la touche **[Suppr]** supprime le caractère se trouvant sous le curseur et déplacent les caractères qui suivent d'une position vers la gauche.

L'appui sur la touche **[Back Space]** déplace le curseur et les caractères qui suivent d'une position vers la gauche. Le caractère qui se trouvait immédiatement à gauche du curseur est effacé.
Si le curseur se trouvait en colonne 1 au moment de l'appui sur la touche **[Back Space]**, l'ensemble de la ligne situé à la position courante du curseur est concaténée à la ligne précédente.

INSERTION LIGNE - [RC]

INSERTION D'UNE LIGNE

L'appui sur la touche **[RC]** en mode insertion positionne le curseur sur la première colonne de la ligne qui suit la ligne courante. Le texte situé à droite du curseur ainsi que le caractère situé sous le curseur sont placés sur la ligne crée. En mode **Ecrasement**, le curseur est positionné sur la première colonne de la ligne qui suit la ligne courante.

SORTIE - [Alt][X]

SORTIE DU SYSTEME DE DEVELOPPEMENT

L'appui sur les touches **[Alt] [X]** provoque le retour au système d'exploitation. Si un fichier en cours d'édition a été modifié et n'a pas été sauvegardé, l'éditeur en propose la sauvegarde.

DEPLACEMENTS

DEPLACEMENTS DANS LA FENETRE D'EDITION

Les touches ayant une action sur le déplacement du curseur sont :

[↑] [↓] [→] [←] : déplacement du curseur d'une position dans la direction indiquée.

[Ctrl] [→] et **[Ctrl] [←]** : positionnement du curseur au début (**[→]**) ou à la fin (**[←]**) du mot suivant ou précédent.

[Fin] : positionnement du curseur à la fin de la ligne courante.

[Home] : positionnement du curseur sur la première colonne de la ligne courante.

[Ctrl] [Fin] : positionnement du curseur sur la dernière ligne de la fenêtre active.

[Ctrl] [Home] : positionnement du curseur sur la première ligne de la fenêtre active.

[▾] : affichage de la page d'écran suivante.

[▴] : affichage de la page d'écran précédente.

[Ctrl] [▾] : positionnement du curseur sur le dernier caractère du fichier en cours d'édition.

[Ctrl] [▴] : positionnement du curseur sur le premier caractère du fichier en cours d'édition.

[RC] : en mode écrasement (insertion inactive), l'appui sur la touche **[RC]** positionne le curseur sur la

première colonne de la ligne qui suit la ligne courante. En mode insertion, l'action de cette touche est sensiblement différente (voir **insertion d'une ligne**).

[Tab] : déplacement du curseur du nombre de colonnes spécifiées dans le paramètre option de tabulation.

UTILISATION DU COMPILATEUR

Le lancement du compilateur se fait en appuyant sur la touche **[F9]** lorsque l'on se trouve sous l'éditeur de texte. Le fichier compilé est celui qui se trouve dans la fenêtre active.

Lors de l'exécution du compilateur, une fenêtre s'affiche au milieu de l'écran. Elle indique le numéro du passage de la compilation en cours et le numéro de la ligne en cours d'analyse.

Il est possible d'arrêter la compilation à tout moment en appuyant sur la touche **[Esc]**.

L'action du compilateur consiste à traduire le programme LECAM source dans le code objet utilisable par l'interpréteur du LECAM.

4 fichiers sont créés :

- **<nom>**.OBJ : ce fichier contient le code objet du programme LECAM.
- **<nom>**.SYM : ce fichier contient la table des symboles du programme compilé.
- **<nom>**.LST : ce fichier contient le listing du programme compilé. Il peut être visualisé par l'éditeur de texte ou être directement imprimé.
- **<nom>**.ADR : ce fichier est utilisé par le débogueur symbolique.

<nom> est le nom du programme source venant d'être compilé.

La structure des fichiers .OBJ et .SYM sont fournis en annexe 1 de ce document.

STRUCTURE D'UN PROGRAMME LECAM

Un programme LECAM se compose :

-d'une déclaration précisant le type de LECAM (1 ou 2) auquel le programme est destiné. Si cette déclaration est absente, le compilateur considère que le programme doit être généré pour un LECAM 1. Les principales différences entre le LECAM 1 et le LECAM 2 concernent l'espace mémoire disponible pour le programme (768 octets pour le LECAM 1, 1892 octets pour le LECAM 2) et certaines particularités d'adressages.

-d'une définition des équivalences (constantes) utilisées.

-d'une suite de programmes délimités par des directives de début et de fin de programme. Ces programmes peuvent eux mêmes contenir des équivalences.

-de commentaires pouvant se trouver dans n'importe qu'elle partie du programme.

Exemple :

```
LECAM1

*
* constantes diverses
*
TABLEEQU45H
VALEUREQU01001100Z
*
* début du programme 1
*
PG1PROGRAM100H

< corps du programme PG1 >

ENDPPG1
*
* début du programme 2
*
PG2PROGRAM$

< corps du programme PG2 >

ENDPPG2
END
```

L'ensemble de ces composants sera nommé **unité de compilation** dans la suite de ce document.

GENERALITES SUR LE FORMAT D'UN PROGRAMME SOURCE

Une ligne de source peut être divisée en quatre zones à la manière d'un langage d'assemblage traditionnel :

-la zone **label**. Un label est un nom symbolique de 8 caractères utiles maximums dont le premier commence obligatoirement en colonne 1. Par défaut la valeur du label vaut la valeur du compteur ordinal. La pseudo-instruction **EQU** permet de lui affecter une autre valeur.

-la zone **instruction**. Pour être interprété comme une instruction et non comme un label, le premier caractère de l'instruction doit être situé en colonne 2 ou dans les colonnes suivantes. Lorsqu'une instruction se trouve sur la même ligne qu'un label, elle doit être séparée de cette dernière par un ou

plusieurs caractères espaces.

-la zone **opérande**. La zone opérande vient immédiatement après la zone instruction dont elle est séparée par un ou plusieurs caractères espaces.

-la zone **commentaire**. Cette zone permet d'écrire un texte libre. Un commentaire commence obligatoirement par le caractère ***** (astérisque). La zone commentaire est facultative mais si elle est présente, elle doit obligatoirement être la dernière d'une ligne source. Toutefois, il est possible de créer une ligne ne comportant qu'un commentaire. Dans ce cas, l'astérisque peut se trouver en première colonne.

Une ligne vide (pas d'instruction, pas de commentaire) ne génère pas d'erreur de syntaxe et est considérée comme un commentaire.

Le compilateur ne fait pas de distinction entre les lettres minuscules ou majuscules.

Exemple :

```
LECAM 1
*
* Exemple de programme pour le compilateur LECAM.
* Le programme est généré pour un LECAM 1.
* Mieux vaut ne pas chercher à quoi il sert...
*
PG1PROGRAM$
*
* Équivalences
*
LECTEQU45
TABLEEQUR05* 1er registre de TABLE

DISP82H
44H,R00
26H,'BONJOUR'
7FH,100H
ABORT0
ADD1,TABLE,R02
ADD7,R01,R07
ADD7,R05,TABLE
ACK
```

```
ENDPPG1
END
```

Conventions de notation

Dans la suite de ce document, on adoptera les conventions de notation suivantes :

<valeur>:les signes **<** et **>** sont utilisés pour désigner une valeur formelle qui devra être remplacée par sa valeur réelle lors de l'écriture du programme.

{... ou ... ou ...}:les signes **{** et **}** sont utilisés pour désignés une liste de valeurs dont une et une seule devra obligatoirement être retenue lors de l'écriture du programme.

[... , ... , ...]:les signes **[** et **]** sont utilisés pour désigner un ou une suite de valeur facultatives.

Les valeurs qui ne sont pas encadrés par l'un de ces signes correspondent aux valeurs réelles telles qu'elles devront être écrites dans un programme.

Exemple :<label>EQU{<valeur> ou <registre> ou $}

<label> devra être remplacé par un nom choisit par l'utilisateur. **<valeur>** peut désigner une valeur numérique, **<registre>** désignera un nom de registre, **$** est une valeur réelle correspondant à la valeur du compteur ordinal. Les **{ }** signifient qu'une des valeurs citées doit être présente dans le programme.

Définition d'un label

Un **label** est un nom mnémonique créé par le concepteur du programme. son premier caractère doit obligatoirement être une lettre alphabétique. Les caractères qui suivent peuvent être des lettres alphabétiques, des chiffres ou le caractère souligné (underscore). Le nombre maximum de caractères utile d'un label est 8. S'il est plus long, les caractères qui suivent le huitième sont ignorés mais ne génère pas d'erreur de compilation.

Les noms qui suivent sont tous corrects :

```
TABLE
R01
ORD_ENT
ORDRE_ENTRANT
```

Les noms qui suivent sont incorrects :

1_A_2

ORDRE_ENTRANT

_OK

Définition d'une instruction

Une instruction est un nom mnémonique ayant une signification propre pour le compilateur. Si l'instruction ne génère pas de code objet pour le LECAM, on parlera de pseudo-instruction. Dans le cas contraire, on parlera d'instruction LECAM.

Dans la majorité des cas, les instructions respectent la présentation telle qu'elle est définie dans les STUCAM.

Définition d'une opérande

Une opérande est une suite de valeurs précisant l'instruction à laquelle elle est associée et/ou d'une suite de valeurs sur lesquelles porte l'instruction.

Exemple : si ADD est un mnémonique pour l'instruction d'addition entre deux nombres, les deux nombres seront les opérandes de ADD.

ADD3,4 * addition de 3 avec 4

Syntaxe des opérandes

Dans CPLECAM, les opérandes peuvent être des mots clés du langage, des valeurs numériques, des chaines de caractères, des mnémoniques définit par le concepteur du programme, des registres du LECAM, etc...

Les mots clés acceptés par le compilateur sont définis instruction par instruction dans le paragraphe **"syntaxe des instructions"**.

Les valeurs numériques peuvent être des nombres écrits en décimal, hexadécimal ou binaire.

Les nombres écrits en décimal sont codés tels-quels (exemple **12** pour douze).

Les nombres écrits en hexadécimal doivent être suivis de la lettre **H**. Ainsi, la valeur **12** en décimal s'écrira **0CH** en hexadécimal.

Si un nombre Hexadécimal commence par un des chiffres de **A** à **F**, il devra être précédé d'un **0** (zéro) afin de ne pas être confondu avec un identificateur. Ainsi, le nombre **CAFE** codé en hexadécimal devra être écrit **0CAFEH**.

Les nombres écrits en binaire doivent être suivis de la lettre **Z**. Ainsi, la valeur **12** en décimal s'écrira **1100Z** en binaire.

Les chaines de caractères doivent être encadrées par des cotes. Si l'on souhaite fournir le texte **12** en opérande, il faudra donc l'écrire **'12'**.

CPLECAM reconnait les noms de registres du LECAM. Ceux-ci doivent débuter par la lettre R suivi de leur numéro codé en hexadécimal. Ainsi le 1er registre du LECAM s'écrira **R00** et le 256ème registre du LECAM s'écrira **RFF**. Cette syntaxe a été adoptée pour permettre de se rapprocher le plus possible des exemples fournis par France-Télécom dans les STUCAM.

SYNTAXE DES PSEUDOS-INSTRUCTIONS

LECTEUR

La directive de définition du type de Lecteur

Syntaxe :LECTEUR<type de lecteur>

<type de lecteur> peut être LECAM1 pour le LECAM 1, LECAM2 pour le LECAM 2 ou GEMTEL pour le lecteur de GEMPLUS. La version 1 de CPLECAM considère, le paramètre GEMTEL comme équivalent à LECAM1.

Exemple :

LECTEURLECAM1

PROGRAM

La directive de début de programme

Syntaxe : <label>PROGRAM{<valeur>,$}

La directive **PROGRAM** permet de définir le début d'un programme LECAM ainsi que son adresse d'implantation en mémoire. Le **<label>** est le nom par lequel le programme est connu. L'opérande **<valeur>** définit l'adresse du début du programme. La valeur **$** correspond à la valeur courante du compteur ordinal.

Un programme LECAM doit obligatoirement se terminer par une directive **ENDP** suivi du nom du programme.

Exemples :

```
PROGPROGRAM100H
PROGPROGRAM$
```

ENDP

La directive de fin de programme

syntaxe :ENDP<label>

La directive **ENPD** indique au compilateur la fin du programme déclaré par PROGRAM. Le label doit être le nom du dernier programme déclaré.

Exemples :

```
PROGPROGRAM100H
...
```

ENDPPROG

EQU

La directive de définition d'une équivalence

Syntaxe :<label>EQU{<valeur>,<registre>,$}

La directive **EQU** permet d'affecter une valeur à un label.
Les opérandes associées à un EQU ne peuvent être que des valeurs numériques sur 16 bits maximum.

Exemples :

TABLEEQU12 * affecte 12 à TABLE.
CLASSEEQU0BCH* affecte 0BCH à CLASSE

Il est possible de rendre équivalent un mnémonique et un nom de registre de façon à rendre le programme plus lisible.

Exemple :

TABLEEQUR32* TABLE sera désormais connu comme le registre R32

Par défaut, un label créé dans le corps du programme prend la valeur du compteur ordinal. Il en est de même pour une définition d'équivalence suivi du signe **$**. Les deux séquences suivantes sont donc équivalentes :

```
LDR0,R50,'A'
ETIQLDR0,R51,'B'
```

est équivalent à

```
LDR0,R50,'A'
ETIQEQU$
```

LDR0,R51,'B'

DATA

La directive de déclaration de données

Syntaxe :DATA<données>

La directive **DATA** permet de stocker des octets dans le corps d'un programme. Cette directive peut par exemple être utilisée pour générer des instructions qui ne seraient pas supportées par la version courante de CPLECAM (évolution du LECAM). Les données peuvent être écrites de la même manière et avec les mêmes contraintes que les valeurs immédiates d'une instruction (voir le chapitre concernant les instructions du LECAM).
Les exemples qui suivent sont tous valides :

DATA010203040506H

Cette directive stockera les octets 01H, 02H, 03H, 04H, 05H et 06H à partir de la valeur courante du compteur ordinal.

DATA'ABCD' 01H 01H

Cette directive stockera les octets 41H, 42H, 43H, 44H, 01H et 01H à partir de la valeur courante du compteur ordinal.

Le nombre d'octets qu'il est possible de générer par une seule directive DATA est limité à 256 octets.

END

La directive de fin de compilation

Syntaxe : END

La fin d'une unité de compilation doit se terminer par la directive **END**. Aucune instruction ou déclaration de programme ne doit se trouver après cette directive.

Inclusion de fichier

Inclusion d'un fichier

syntaxe :<nom du fichier>

Les signes **<** et **>** doivent encadrer le nom du fichier à inclure.

Exemple :

<inst.cst>

Remarque: CPLECAM est fourni avec un fichier de définition des constantes nommé **inst.cst** Ce fichier définit les valeurs des paramètres **type**, **mode**, etc... de certaines instructions. Les exemples de codages fournis dans le chapitre *syntaxe des instructions* supposent que ce fichier à été inclu en début de programme.

SYNTAXE DES INSTRUCTIONS

Préambule

Le mode de notation adopté dans ce chapitre reprend défini dans les STUCAM. Les exemples fournis correspondent à un codage accepté par CPLECAM. On suppose que le fichier **inst.cst** définissant la valeur des paramètres de certaines instructions a été inclue en début de programme.

Adressage

Le LECAM accepte des adressages en mode absolu ou en mode relatif. Dans une instruction ou intervient une adresse de branchement, une adresse relative devra être précédée du signe **@** . Une adresse absolue sera noté sans autre attribut que sa valeur.

Exemple :**GOTOBR1**

Cette instruction provoque un branchement inconditionnel à l'adresse absolue BR1.

GOTO@BR1

Cette instruction provoque un branchement inconditionnel à l'adresse relative BR1.

Références en avant et en arrière

Les labels intervenant dans une instruction peuvent toujours être définis "en arrière" (la valeur du label est connue au moment ou l'instruction est compilée). Seules, les adresses et les valeurs immédiates (cf STUCAM) des instructions acceptent les références en avant (la valeur du label n'est pas connue au moment ou l'instruction est compilée.

Ainsi, les valeurs intervenant dans le "mode" d'une instruction devront être obligatoirement définies avant que l'instruction ne soit compilée. Il en est de même en cas de renommage de registres. Par contre, une adresse de branchement pourra être définie en avant ou en arrière.

Opérateurs

CPLECAM accepte les opérateurs **+** (addition), **-** (soustraction), **&** (ET logique) et **!** (OU logique).

Ces opérateurs peuvent intervenir dans une définition de longueur, de mode, d'une valeur immédiate et d'une équivalence.

Les additions et les soustractions ne portent que sur des valeurs non signées.

***** actuellement, uniquement dans les longueurs et les modes *****

ABORT

Action:Signalisation d'erreur

Syntaxe:**ABORTmode**

Le paramètre **mode** contient le mode de l'instruction ainsi que le code d'abort devant être renvoyé à l'application.

Exemples :

ABORTADR_ERR+03H

Cette instruction renvoie l'adresse de la dernière instruction ayant provoqué une erreur traitée par ONERR ainsi que le code d'erreur 03H

ABORT04H

Cette instruction revoie la valeur des registres R00 et R01 ainsi que le code d'erreur 04H

ACK

Action:Dépilage d'adresse

Syntaxe:**ACK**

ADD

Action:Addition

Syntaxe:**ADDlg,RD,RS**

ADDlg,RD,VI

Lg est la longueur du bloc de données à additionner. Selon le mode de l'instruction, le paramètre à additionner au bloc débutant au registre RD est un bloc débutant au registre RS ou une valeur immédiate VI.

Exemples:

ADD0+ADD_REGISTRE,R10,R12

Cette instruction additionne le contenu des registres R10 et R12. Le résultat est stocké dans R10.

ADD2+ADD_IMMEDIAT,R13,010203H

Cette instruction additionne le contenu du bloc de registres R13 à R15 à la valeur 010203H. Le résultat est stocké dans le bloc de registre R13 à R15.

AND

Action:ET logique
Syntaxe:**ANDLg,RD,RS**
ANDLg,RD,VI

Lg est la longueur du bloc de données sur lequel porte le ET logique. Selon le mode de l'instruction, le paramètre sur lequel porte le ET est un bloc débutant au registre RS ou une valeur immédiate VI. Le résultat est stocké dans le bloc débutant au registre RD.

Exemples:

AND0+AND_REGISTRE,R10,R12

Cette instruction effectue un "et logique" entre le contenu des registres R10 et R12. Le résultat est stocké dans R10.

AND2+AND_IMMEDIAT,R13,010203H

Cette instruction effectue un "et logique" entre le contenu du bloc de registres R13 à R15 à la valeur 010203H. Le résultat est stocké dans le bloc de registre R13 à R15.

BRPR

Action:Branchement sur table de profil

Syntaxe:**BRPRRTP**

RTP doit être le registre contenant le début dela table de profil(s).

Exemple:

BRPRR90

Branchement sur la table de profil(s) implantée à partir de l'adresse R90.

BRST

Action:Branchement sur table d'état
Syntaxe:**BRSTRTE**

RTE doit être le registre contenant le début de la table d'état.

Exemple:

BRSTR8B

Chargement de la table d'état à partir du registre R8B.

CALL

Action:Appel de sous-programme
Syntaxe:**CALL Etiq**

Etiq est l'adresse d'un sous programme. Si le branchement doit être effectué en relatif, Etiq doit être précédé d'un **@** .

Exemple:

CALL0009H

Appel du sous programme situé à l'adresse 9 de la mémoire.

```
ETIQEQU$
...
CALLETIQ
```

Appel du sous programme Etiq dont l'adresse est défini par la valeur du compteur ordinal au moment de la compilation.

COMPUTE

Action:Combinaison d'ordre entrant et sortant
Syntaxe:**COMPUTEINS1,RA,LgDE,RS,INS2,LgDS,RTE**

INS1 est un ordre entrant d'une carte, RA est un bloc de registre contenant le paramètre "adresse" de l'instruction, LGde est le paramètre "longueur" de l'instruction, RS est le registre contenant le début des "données" pour l'instruction.

INS2 est un ordre sortant d'une carte, LgDS est la longueur des données attendues, RTE est le registre contenant le début d'une table d'état

Exemple:

```
CALCULEQU0D0H* ordre entrant
RESULTEQU0B0H* ordre sortant
ADRESSEEQURA3* adresse carte
DONNEESEQUR31* données
RTEEQUR8B* table d'état

COMPUTECALCUL,ADRESSE,8,DONNEES,RESULT,8,RTE
```

Demande d'exécution d'un calcul (code instruction CALCUL) sur les données contenues dans DONNEES à l'adresse ADRESSE. Demande de résultat (instruction RESULT) d'une longueur 8. La table d'état utilisée est celle située en RTE.

CVAD

Action:Conversion ASCII/Hexadécimale

Syntaxe:**CVADLg,RD,RS**

Lg est la longueur du bloc à convertir, RD est le numéro du premier registre recevant le résultat, RS est le numéro du premier registre du bloc à convertir.

Exemple:

CVAD2,R50,R10

Demande de conversion de 3 octets dont le premier est contenu dans R10. Le résultat est stocké à partir de R50.

CVBD

Action:Conversion binaire/DCB

Syntaxe:**CVDBLg,RD,RS**

Lg est la longueur du bloc à convertir, RD est le numéro du premier registre recevant le résultat, RS est ke numéro du premier registre du bloc à convertir.

Exemple:

CVDB1,R10,RA0

Conversion en DCB du bloc de registre de longueur 2 et d'adresse RA0. Le résultat est stocké dans R10, R11 et R12.

CVDA

Action:Conversion hexadécimale/ASCII

Syntaxe:**CVDALg,RD,RS**

Lg est la longueur du bloc à convertir, RD est le numéro du premier registre recevant le résultat, RS est ke numéro du premier registre du bloc à convertir.

Exemple:

CVDA2,R10,R50

Conversion de caractères hexadécimaux contenus dans R51, R51 et R52 en ASCII. Le résultat est stocké dans les registres R10 à R15.

CVDB

Action:Conversion DCB/binaire
Syntaxe:**CVDBLg,RD,RS**

Lg est la longueur du bloc à convertir, RD est le numéro du premier registre recevant le résultat, RS est ke numéro du premier registre du bloc à convertir.

Exemple:

CVDB2,R10,R50

Conversion en binaire des données en DCB contenu dans R50, R51 et R52. Le résultat est stocké dans R10 et R11.

DIAL

Action:Dialogue sensible
Syntaxe:**DIAL<paramètres VALID>,<paramètres DISP>**

DICH

Action:Recherche dichotomique

Syntaxe:**DICHINS,RBd,RBf,RTP,RTE**

INS est le code de l'instruction d'une carte, RBd est le numéro du premier registre contenant la borne de début de recherche, RBf est le numéro du premier registre contenant la borne de fin de recherche, RTP est le registre contenant le début de la table de profil(s), RTE est le registre contenant le début de la table d'état.

Exemple:

LECTUREEQU0B0H* instruction de lecture
DEBUTEQUR10* adresse début recherche
FINEQUR20* adresse fin recherche
PROFILEQUR80* table de profil(s)
ETATEQUR90* table d'état

DICHLECTURE,DEBUT,FIN,PROFIL,ETAT

DISP

Action:Envoi d'un message sur l'écran du Minitel
Syntaxe:**DISPLg,Mode,{RE ou VI ou PG}* mode 1**

DISPNb,Mode* mode 2
Lg,Mode,{RE ou VI ou PG}
...
Lg,Mode,{RE ou VI ou PG}

2 syntaxes sont possibles selon la nature du message à transmettre déterminé par le "type" du paramètre "mode".

S'il n'y a pas de chaînage, l'instruction se présente selon le mode 1. Si le message est du type registre, le paramètre RE est le premier registre contenant le message. Si le message appartient à une zone programme, PG est l'adresse de cette zone. Si le message est du type "valeur immédiate", VI est cette valeur. Dans tous les cas, Lg est la longueur du message.

S'il y a chaînage, Nb est le nombre d'éléments chaînés. Chaque éléments est constitué d'un paramètre "mode" et du message dont la nature est déterminé par le paramètre "mode" (voir ci-

dessus).

Exemple:

```
DISP5,DISP_IMMEDIAT+DISP_RAZ,'ABCD' 0A0DH
```

Envoi du message contenue dans la valeur immédiate avec remise à zéro préalable. La valeur hexadécimale du message envoyé sera 41H, 42H, 43H, 44H, 0AH, 0DH.

```
PRG1EQU$
...

DISP1,DISP_CHAINE
3,DISP_REGISTRE,R00
2,DISP_PROGRAMME,PRG1
```

Envoi du message chainé composé de deux éléments. Le premier comporte 4 registres dont commençant par R00. Le deuxième comporte 3 octets de programme dont l'adresse est définie par PRG1.

DISPS
DISPZ

Action:Modification de la rangée dédiée du Minitel
Syntaxe:**DISPS**
DISPZ

ENTER

Action:Saisie clavier
Syntaxe:**ENTERLg,Mode,RD,Etiq**

Lg est la longueur des données à saisir au clavier hors touches de fonctions. Mode définit le type d'écho, RD est le registre à partir duquel les données sont stockées, Etiq est une adresse de

branchement.

Exemple:

```
DONNEESEQUR01
...

ENTER10,ECHO_NORMAL,DONNEES,BR1
...

BR1EQU$
...
```

Saisie de 11 caractères avec écho. Les données saisies sont stockées à partir de R01. Branchement en BR1 ou continuation en séquence selon la touche de fonction utilisée pour la fin de la saisie.

GOTO

Action:Branchement inconditionnel
Syntaxe:**GOTOEtiq**

Etiq est l'adresse de branchement du GOTO.

Exemple:

```
BR1EQU$
...

GOTO@BR1
```

Branchement inconditionnel à l'adresse BR1 (adressage relatif).

```
GOTOBR2
...

BR2LDR1,R00,'AB'
```

Branchement inconditionnel à l'adresse BR2 (adressage absolu).

```
GOTO0009H
```

Branchement inconditionnel à l'adresse 9 (adressage absolu).

IF

Action:Branchement conditionnel
Syntaxe:**IFLg,Mode,Etiq,RS,{RR ou VI}**
IFLg,Mode,Etiq,RS,{RR,masque}
IFLg,Mode,Etiq,RS,{VI,masque}

Lg est la longueur du bloc de données à comparer. Mode définit le type de comparaison ainsi que la nature du bloc intervenant dans la comparaison. Etiq est l'adresse de branchement. RS est le registre où débute le bloc de données à tester. Selon la valeur du Mode, RR ou VI contient le bloc de données de référence. Toujours selon la valeur du Mode, un masque est associé à RR ou VI.

Exemple:

```
IF3,IF_LOGIQUE+IF_REGISTRE,BR1,R0F,R80,0FFFFFFFEH
...

BR1EQU$
```

Comparaison de type logique sur un bloc de données débutant en R0F par rapport à une valeur de référence débutant en R80. La valeur du masque est 0FFFFFFFEH. L'adresse de branchement est BR1.

```
BR2EQU$
...

IF1,IF_EGAL+IF_IMMEDIAT+IF_INVERSE,BR2,R0F,R80
```

Comparaison arithmétique inverse portant sur l'inégalité entre le bloc de données débutant en R0F avec la valeur de référence débutant en R80. La longueur des données est de deux octets. En cas

d'inégalité, il y a branchement à l'adresse BR2.

INPUT

Action:Envoi d'un ordre sortant

Syntaxe:**INPUTINS,RA,LgDS,RTE**

INS est le code de l'ordre sortant d'une carte. RA est le bloc de registre contenant le paramètre "adresse" de l'ordre, LgDS est la longueur réelle des données sortantes, RTE est le registre contenant le début d'une table d'état.

Exemple:

```
ADRESSEEQUR81*adresse carte
ETATEQUR8B*table d'état
LECTUREEQU0B0H*code instruction lecture
...

INPUTLECTURE,ADRESSE,10H,ETAT
```

Lecture de 16 octets à partir de l'adresse contenue dans les registres R81 et R82 en utilisant la table d'état définie à partir de R8B.

Le codage ci-dessous est équivalent :

```
INPUT0B0H,R81,10H,R8B
```

IOR

Action:OU logique

Syntaxe:**IORLg,RD,RS**

IORLg,RD,VI

Lg est la longueur du bloc de données sur lequel porte le OU logique. Selon le mode de l'instruction, le paramètre sur lequel porte le OU est un bloc débutant au registre RS ou une valeur immédiate VI. Le résultat est stocké dans le bloc débutant au registre RD.

Exemples:

IOR0+IOR_REGISTRE,R10,R12

Cette instruction effectue un "OU logique" entre le contenu des registres R10 et R12. Le résultat est stocké dans R10.

IOR2+IOR_IMMEDIAT,R13,010203H

Cette instruction effectue un "OU logique" entre le contenu du bloc de registres R13 à R15 à la valeur 010203H. Le résultat est stocké dans le bloc de registre R13 à R15.

LDR

Action:Chargement de registre(s)
Syntaxe:**LDRLg,RD,VI**

Lg est la longueur des données immédiates VI à charger à partir du registre RD.

Exemple:

LDR10,R10,'BONJOUR !' 0D0AH

Chargement du texte ASCII 'BONJOUR' suivi des octets 0DH et 0AH dans les registres débutant en R10.

MD

Action:Passage en mode distant
Syntaxe:**MD**

MHT

Action:Mise hors tension d'une carte

Syntaxe:MHT

MST

Action:Mise sous tension d'une carte
Syntaxe:**MSTRTE**

RTE est un registre contenant le début de la table d'état.

Exemple:

```
ETATEQUR80* table d'état
...

MSTETAT
```

Mise sous tension d'une carte en utilisant la table d'état implantée à partir du registre R80.

NEG

Action:Complément à deux
Syntaxe:**NEGLg,RD**

Lg est la longueur des données sur lesquelles porte l'opération de négation. RD est le registre à partir duquel débute le bloc de données.

Exemple:

```
NEG1,R10
```

Complément à 2 du bloc de registres R10 et R11. Le résultat est stocké dans R10 et R11.

NKEY

Action:Pas de clé disponible
Syntaxe:**NKEY**

NOP

Action:Pas d'opération
Syntaxe:**NOP**

NOT

Action:Complément à un
Syntaxe:**NOTLg,RD**

Lg est la longueur des données sur lesquelles porte la négation. RD est le registre à partir duquel débutent le bloc de données.

Exemple:

NOT2,R51

Complément à 1 des trois registres R51, R52, R53. Le résultat est stocké dans ces mêmes registres.

ONERR

Action:Débranchement sur erreur
Syntaxe:**ONERREtiq**

Etiq est l'adresse de branchement en cas d'erreur lors de l'exécution d'une instruction.

Exemple:

BRAEQU$

...

ONERRBR1

Branchement à l'adresse BR1 en cas d'erreur.

OUTPUT

Action:Envoi d'un ordre entrant
Syntaxe:**OUTPUTINS,RA,LgDE,RS,RTE**

INS est le code de l'ordre entrant d'une carte, RA est le bloc de registre contenant l'adresse de la carte, LgDE est la longueur des données entrantes, RS est le registre contenant le début de données sortantes, RTE est l'adresse d'une table d'état.

Exemple:

ECREQU0D0H*ordre d'écriture
ADRESSEEQUR81*adresse dans la carte
DATAEQUR45*données sortantes
ETATEQUR8B*table d'état
...

OUTPUTECR,ADRESSE,4,DATA,ETAT

Écriture à partir de l'adresse carte contenue dans R81 et R82 des 4 octets contenus dans R45, R46, R47 et R48 en utilisant la table d'état définie à partir du registre R8B.

PAD

Action:Initialisation de registres
Syntaxe:**PADLg,RD,VI**

Lg est la longueur du bloc de registre à initialiser, RD est le premier registre destination, VI est la valeur immédiate à propager dans les registres destinations (1 octet).

Exemple:

PAD255,R00,0FFH

Initialisation des 256 registres de la mémoire du LECAM (R00 à RFF) avec la valeur 0FFH.

RDNXT

Action:lecture carte avec incrémentation automatique
Syntaxe:**RDNXTINS,RA,RTE**

INS est le code de l'ordre d'une carte, RA est le bloc de registre contenant l'adresse de la carte, RTE est le registre contenant le début de la table d'état.

Exemple:

LECTUREEQU0B0H*ordre de lecture
ADRESSEEQUR20*adresse de lecture
ETATEQUR2A*table d'état
...

RDNXTLECTURE,ADRESSE,ETAT

Lecture dans la carte du nombre d'octet défini par le registre réservé "longueur/pas". l'adresse carte est contenue dans le bloc de registres R20 et R21. L'adresse de la table d'état débute en R2A.

RDPRV

Action:Lecture carte avec décrémentation automatique
Syntaxe:**RDPRVINS,RA,RTE**

INS est le code de l'ordre d'une carte, RA est le bloc de registre contenant l'adresse de la carte, RTE est le registre contenant le début de la table d'état.

Exemple:

LECTUREEQU0B0H*ordre de lecture

ADRESSEEQUR20*adresse de lecture

ETATEQUR2A*table d'état

...

RDPRVLECTURE,ADRESSE,ETAT

Lecture dans la carte du nombre d'octet défini par le registre réservé "longueur/pas". l'adresse carte est contenue dans le bloc de registres R20 et R21. L'adresse de la table d'état débute en R2A.

READ

Action:Lecture dans la carte

Syntaxe:**READINS,RA,RTE**

INS est le code de l'ordre d'une carte, RA est le bloc de registre contenant l'adresse de la carte, RTE est le registre contenant le début de la table d'état.

Exemple:

LECTUREEQU0B0H*ordre de lecture

ADRESSEEQUR20*adresse de lecture

ETATEQUR2A*table d'état

...

READLECTURE,ADRESSE,ETAT

Lecture dans la carte du nombre d'octet défini par le registre réservé "longueur/pas". l'adresse carte est contenue dans le bloc de registres R20 et R21. L'adresse de la table d'état débute en R2A.

REP

Action:Répétition d'une séquence d'instruction

Syntaxe:**REPRC,Etiq**

RC est un registre compteur d'itération, Etiq est l'adresse de branchement sur le début du bloc d'instruction à répéter. Cette instruction est équivalente à un **repeat...until**.

Exemple:

```
LDR0,R10,5
REPETNOP
...
REPR10,REPET
```

5 répétitions de la séquence comprise entre REPET et REP. Le compteur d'itération est le registre R10.

RET

Action:Retour de sous-programme
Syntaxe:**RET**

RKEY

Action:Initialisation du GOC
Syntaxe:**RKEYRS**

RS est le premier registre contenant le bloc d'initialisation du GOC.

Exemple:

```
RKEYR12
```

Initialisation du GOC avec les huits octets implantés à partir du registre R12.

RL

Action:Rotation à gauche
Syntaxe:**RLLg,compte,RD**

Lg est le nombre de registres concernés par le décalage, compte est le nombre de bits à décaler, RD est le premier registre contenant le bloc de données à décaler.

Exemple:

RL2,5,R10

Décalage du bloc de données contenu dans R10, R11 et R12 de 6 positions. Le résultat est rendu dans les mêmes registres.

RR

Action:Rotation à droite
Syntaxe:**RRLg,compte,RD**

Lg est le nombre de registres concernés par le décalage, compte est le nombre de bits à décaler, RD est le premier registre contenant le bloc de données à décaler.

Exemple:

RR2,5,R10

Décallage du bloc de données contenu dans R10, R11 et R12 de 6 positions. Le résultat est rendu dans les mêmes registres.

SCAN

Action:Recherche séquentielle

Syntaxe:**SCANINS,RBd,RBf,RTP,RTE**

INS est le code de l'ordre de lecture d'une carte, RBd est le bloc de registres contenant la borne de début de recherche, RBf est le bloc de registres contenant le bloc de fin de recherche, RTP est le registre contenant le début de la table de profil(s), RTE est le registre contenant le début de la table d'état.

Exemple:

```
LECTUREEQU0B0H*ordre de lecture
DEBUTEQUR10*borne de début
FINEQUR40*borne de fin
PROFILEQUR33*table de profil(s)
ETATEQUR90*table d'état
...

LDR7,PROFIL,902E02FFFF800900H
SCANLECTURE,DEBUT,FIN,PROFIL,ETAT
...
```

Recherche séquentielle entre l'adresse de début contenue dans les registres R10 et R11 et l'adresse de fin contenue dans les registres R40 et R41. La table de profil démarre à partir du registre R33, la table d'état à partir du registre R90.

SCR

Action:Envoi du caractère CR

Syntaxe:**SCR**

SENDC
SEND

Action:Envoi de message au serveur

Syntaxe:**SENDLg,mode,{RE ou EP ou VI ou RG}* mode 1**

SENDNb,mode* mode 2

Lg,Mode,{RE ou EP ou VI ou RG}

...

Lg,Mode,{RE ou EP ou VI ou RG}

2 syntaxes sont possibles selon la nature du message à transmettre déterminé par le "type" du paramètre "mode".

S'il n'y a pas de chaînage, l'instruction se présente selon le mode 1. Si le message est du type registre, le paramètre RE est le premier registre contenant le message. Si le message appartient à une zone programme, EP est l'adresse de cette zone. Si le message est du type "valeur immédiate", VI est cette valeur, si le message est du type "tampon de rangement", RG est implicite et correspond aux registres R04 et R05. Dans tous les cas, Lg est la longueur du message.

S'il y a chaînage, Nb est le nombre d'éléments chaînés. Chaque élément est constitué d'un paramètre "mode" et du message dont la nature est déterminé par le paramètre "mode" (voir ci-dessus).

Exemple:

```
SEND5,SEND_IMMEDIAT,'ABCD' 0A0DH
```

Envoi du message contenue dans la valeur immédiate avec remise à zéro préalable. La valeur hexadécimale du message envoyé sera 41H, 42H, 43H, 44H, 0AH, 0DH.

```
PRG1EQU$
...

SEND2,SEND_CHAINE
3,SEND_REGISTRE,R00
2,SEND_PROGRAMME,PRG1
0,SEND_RANGEMENT
```

Envoi du message chainé composé de deux éléments. Le premier comporte 4 registres dont commençant par R00. Le deuxième comporte 3 octets de programme dont l'adresse est définie par PRG1.

SL

Action:Décalage à gauche
Syntaxe:**SLLg,bit,compte,RD**

Lg est le nombre de registres concernés par le décalage, compte est le nombre de bits à décaler, RD est le premier registre contenant le bloc de données à décaler. Bit est la valeur du bit de remplissage.

Exemple:

SL2,0,5,R10

Décalage du bloc de données contenu dans R10, R11 et R12 de 6 positions. Le résultat est rendu dans les mêmes registres.

SR

Action:Décalage à droite
Syntaxe:**SRLg,bit,compte,RD**

Lg est le nombre de registres concernés par le décalage, compte est le nombre de bits à décaler, RD est le premier registre contenant le bloc de données à décaler. Bit est la valeur du bit de remplissage.

Exemple:

SR2,0,5,R10

Décalage du bloc de données contenu dans R10, R11 et R12 de 6 positions. Le résultat est rendu dans les mêmes registres.

STOP

Action:Arrêt
Syntaxe:**STOP**

SUB

Action:Soustraction

Syntaxe:**SUBlg,RD,RS**

SUBlg,RD,VI

Lg est la longueur du bloc de données à soustraire. Selon le mode de l'instruction, le paramètre à soustraire au bloc débutant au registre RD est un bloc débutant au registre RS ou une valeur immédiate VI.

Exemples:

SUB0+SUB_REGISTRE,R10,R12

Cette instruction soustrait le contenu des registres R10 et R12. Le résultat est stocké dans R10.

SUB2+SUB_IMMEDIAT,R13,010203H

Cette instruction soustrait le contenu du bloc de registres R13 à R15 à la valeur 010203H. Le résultat est stocké dans le bloc de registre R13 à R15.

SWB

Action:Autorisation d'écriture

Syntaxe:**SWB**

TCP

Action:Rangement sélectif du tampon carte

Syntaxe:**TCPparam**

Param sont les paramètres **s**, **a**, **L**, **d** de l'instruction.

Exemple:

TCP0001Z
ou
TCPTCP_DONNEES

Stockage des données seulement.

TCPTCP_DONNEES+TCP_Ai+TCP_L+TCP_ETAT

Stockage de toutes les données.

TPR

Action:Dépilage du tampon de paramètres
Syntaxe:**TPRLg,RD**

Lg est le nombre d'octets à dépiler, RD est le premier registre devant contenir le résultat.

Exemple:
TPR2,R40

Dépilage de 3 octets du tampon de paramètres et stockage dans les registres R40, R41 et R42.

TR

Action:Recopie d'un bloc de registres
Syntaxe:**TRRLg,RRD,RRS**

RLg est le registre contenant la longueur du bloc de registres à recopier, RRD est le premier registre de la destination, RRS est le premier registre source.

Exemple:

TRR10,R20,R40

Recopie des registres à partir de R40 dans R20. R10 contient le nombre d'octets à recopier.

TRP

Action:Empilage de registres dans le tampon de rangement
Syntaxe:**TRPLg,RS**

Lg est le nombre d'octets à empiler, RS est le premier registre contenant les données à empiler.

Exemple:

TRP2,R14

Empilage des 3 registres R14, R15 et R16 dans le tampon de rangement.

TRR

Action:Recopie d'un bloc de registres
Syntaxe:**TRRLg,RD,RS**

Lg est la longueur du bloc de registres à recopier, RS est le premier registre du bloc de données à recopier, RD est le premier registre du bloc récepteur.

Exemple:

TRR2,R10,R50

Recopie des registres R50, R51 et R52 dans R10, R11 et R12.

VALID

Action:Saisie clavier contrôlée
Syntaxe:**VALIDmode,RReF,Etiq**

Mode définit le type de l'écho durant la saisie, RReF est le premier registre d'un bloc de données contenant une chaîne de référence, Etiq est une adresse de branchement.

Exemple:

```
BR1EQU$
...

VALIDECHO_NORMAL,RAB,BR1
```

Saisie avec Echo normal, la chaîne de référence débute à partir du registre RAB, BR1 est l'adresse de branchement.

WRITE

Action:Écriture dans la carte
Syntaxe:**WRITEINS,RA,RTE**

INS est le code de l'ordre d'une carte, RA est le bloc de registre contenant l'adresse de la carte, RTE est le registre contenant le début de la table d'état.

Exemple:

```
ECREQU0B0H*ordre d'écriture
ADRESSEEQUR20*adresse d'écriture
ETATEQUR2A*table d'état
...

WRITEECRITURE,ADRESSE,ETAT
```

Écriture dans la carte du nombre d'octet défini par le registre réservé "longueur/pas". l'adresse carte est contenue dans le bloc de registres R20 et R21. L'adresse de la table d'état débute en R2A.

WRNXT

Action:Écriture carte avec incrémentation automatique
Syntaxe:**WRNXTINS,RA,RTE**

INS est le code de l'ordre d'une carte, RA est le bloc de registre contenant l'adresse de la carte, RTE est le registre contenant le début de la table d'état.

Exemple:

```
ECRITUREEQU0B0H*ordre d'écriture
ADRESSEEQUR20*adresse d'écriture
ETATEQUR2A*table d'état
...

WRNXTECRITURE,ADRESSE,ETAT
```

Écriture dans la carte du nombre d'octet défini par le registre réservé "longueur/pas". l'adresse carte est contenue dans le bloc de registres R20 et R21. L'adresse de la table d'état débute en R2A.

WRPRV

Action:Écriture carte avec décrémentation automatique
Syntaxe:**WRPRVINS,RA,RTE**

INS est le code de l'ordre d'une carte, RA est le bloc de registre contenant l'adresse de la carte, RTE est le registre contenant le début de la table d'état.

Exemple:

```
ECRITUREEQU0B0H*ordre d'écriture
ADRESSEEQUR20*adresse d'écriture
ETATEQUR2A*table d'état
...

WRPRVECRITURE,ADRESSE,ETAT
```

Écriture dans la carte du nombre d'octet défini par le registre réservé "longueur/pas". l'adresse carte est contenue dans le bloc de registres R20 et R21. L'adresse de la table d'état débute en R2A.

XCH

Action:Échange de contenu
Syntaxe:**XCHRD1,RD2**

RD1 est la paire de registre à échanger avec RD2.

Exemple:

XCHR00,R02

Echange des registres R00 et R01 avec R02 et R03.

XOR

Action:OU exclusif
Syntaxe:**XORLg,RD,RS**
XORLg,RD,VI

Lg est la longueur du bloc de données sur lequel porte le OU exclusif. Selon le mode de l'instruction, le paramètre sur lequel porte le ET est un bloc débutant au registre RS ou une valeur immédiate VI. Le résultat est stocké dans le bloc débutant au registre RD.

Exemples:

XOR0+XOR_REGISTRE,R10,R12

Cette instruction effectue un "OU exclusif" entre le contenu des registres R10 et R12. Le résultat est stocké dans R10.

XOR2+XOR_IMMEDIAT,R13,010203H

Cette instruction effectue un "OU exclusif" entre le contenu du bloc de registres R13 à R15 à la valeur 010203H. Le résultat est stocké dans le bloc de registre R13 à R15.

UTILISATION DU DEBOGUEUR

GENERALITES

On accède au débogueur symbolique à partir de l'éditeur de texte en appuyant sur la touche **[F8]**.

L'aspect de l'écran est le même que celui de l'éditeur à la différence près qu'il y a systématiquement deux fenêtres. La première (fenêtre "fichier") est réservée à l'affichage du fichier en cours de mise au point, la deuxième (fenêtre "trace") sert à l'affichage des traces lors des dialogues avec le LECAM.

-le bandeau situé sur la première ligne de l'écran rappelle les principales touches de fonctions actives pour le débogueur,

-la deuxième ligne fournit des indications dynamiques correspondant à l'état courant du débogueur :

♦**Lig** : n° de ligne courant où se trouve le curseur,

♦**Col** : n° de colonne courant ou se trouve le curseur,

♦**Débogueur** : cet indicateur précise que l'on se trouve sous le débogueur du système de développement. Son autre valeur possible est **Editeur**.

♦**<nom de fichier>** : il s'agit du nom de fichier en cours de mise au point. S'il n'y a pas de fichier chargé, la valeur de <nom de fichier> est **NONAME**.

C'est également cette ligne qui est utilisée lors de l'affichage de messages d'erreurs ou d'informations.

Le rôle du débogueur symbolique est de faciliter la mise au point d'un programme LECAM. Le principe général est le suivant :

lors de la compilation d'un fichier source (.LEC), le compilateur génère différents fichiers dont certains vont être utilisés par le débogueur symbolique. Il s'agit du fichier .LST (fichier listing), du fichier .ADR (correspondance entre le .LST et le code objet) et du fichier .OBJ (code objet) qui contient le programme compilé.

Lors de la mise au point d'un programme LECAM, l'utilisateur doit charger le fichier .LST correspondant. Le débogueur vérifie l'existence des fichiers .OBJ et .ADR. S'ils existent, le fichier .LST est chargé dans la fenêtre "fichier" du débogueur.

L'utilisateur peut passer de la fenêtre "fichier" à la fenêtre "trace" de la même façon que le passage d'une fenêtre à l'autre dans l'éditeur de texte. Il peut utiliser les mêmes touches de fonctions pour le déplacement du curseur.

A partir du moment ou un fichier est chargé, il est possible d'émettre des consignes au LECAM, donc, en particulier, de charger un ou plusieurs programmes et d'en demander l'exécution.

La pose d'un point d'arrêt se fait en téléchargeant une instruction ABORT à l'adresse ou le programme LECAM doit être interrompu. Pratiquement, il suffit à l'utilisateur de positionner le curseur sur la ligne correspondant à l'endroit où doit être posé le point d'arrêt et d'appuyer sur la touche **[F2]**. La ligne apparaitra alors en blanc sur fond rouge (ou en surbrillance pour un moniteur monochrome). La dépose d'un point d'arrêt se fait selon le même principe.

Lorsque l'utilisateur demande l'exécution de son programme, les points d'arrêts sont téléchargés dans la mémoire du LECAM et l'exécution commence à l'adresse spécifiée par l'utilisateur.

Lorsque le LECAM arrive sur une instruction ABORT, l'interpréteur LECAM renvoie l'indication d'ABORT au débogueur ainsi que son numéro. C'est grâce à ce numéro que le débogueur peut retrouver l'adresse où s'est arrête l'interpréteur et le signaler ainsi à l'utilisateur.

Le point d'arrêt ayant provoqué l'arrêt de l'interpréteur est alors retiré et l'instruction qui avait été écrasé par l'ABORT est téléchargée (en fait le retrait du point d'arrêt et la réinitialisation de l'instruction se fait à la prochaine demande d'exécution du programme LECAM).

Le débogueur utilise les ABORT 0 à 9 pour ses points d'arrêt. *Il est donc conseillé de ne pas utiliser ces numéros d'ABORT dans un programme LECAM.*

***** à compléter *****

ANNEXE 1 - FICHIER DE CONSTANTES

Le fichier de constante **INST.CST** est livré avec l'environnement de développement CPLECAM. A l'installation, il est copié dans le même répertoire que celui de CPLECAM. Le listing de ce fichier est fourni ci-dessous :

```
            * ------------------------------------------- *
            * définition des options pour les instructions *
            *                                             *
            * fichier inst.cst. A.G.I.  1991              *
            * ------------------------------------------- *
*
*               ABORT
*               -----
*
ADR_ERR         EQU     80H     * adresse dernière instruction traitée par ONERR
*
*               ADD
*               ---
ADD_IMMEDIAT    EQU     08H     * opérande "valeur immédiate"
ADD_REGISTRE    EQU     00H     * opérande "registre"
*
*
*               AND
*               ---
AND_IMMEDIAT    EQU     08H     * opérande "valeur immédiate"
AND_REGISTRE    EQU     00H     * opérande "registre"
*
*               DISP
*               ----
*
*               mode de DISP
*
DISP_CHAINE     EQU     00H     * message chainé
DISP_IMMEDIAT   EQU     20H     * type immédiat
DISP_REGISTRE   EQU     40H     * type registre
DISP_PROGRAMME  EQU     60H     * type programme
*
*               ENTER / VALID
*               -------------
*
ECHO_NORMAL     EQU     80H     * echo normal pour mode de ENTER
ECHO_BROUILLE   EQU     00H     * echo brouillé
*
*               IF    mode = type + S + M
*               -------------------------
*               type du IF
*
IF_LOGIQUE      EQU     00H     * comparaison logique
IF_INFERIEUR    EQU     40H     * comparaison arithmétique inférieure
IF_EGAL         EQU     80H     * égalite
IF_SUPERIEUR    EQU     0C0H    * comparaison arithmétique supérieure
*
*               S du if
*
IF_DIRECT       EQU     00H     * sens direct pour la condition
IF_INVERSE      EQU     20H     * sens inverse pour la condition
*
*               M du if
*
IF_IMMEDIAT     EQU     10H     * référence immédiate
IF_REGISTRE     EQU     00H     * référence registre
*
*               IOR
*               ---
IOR_IMMEDIAT    EQU     08H     * opérande "valeur immédiate"
IOR_REGISTRE    EQU     00H     * opérande "registre"
*
*               SEND
*               ----
*
```

```
*                type
*
SEND_CHAINE      EQU       00H     * message chainé
SEND_IMMEDIAT    EQU       20H     * type immédiat
SEND_REGISTRE    EQU       40H     * type registre
SEND_PROGRAMME   EQU       60H     * type programme
*
*                SUB
*                ---
SUB_IMMEDIAT     EQU       08H     * opérande "valeur immédiate"
SUB_REGISTRE     EQU       00H     * opérande "registre"
*
*                TCP       S+A+L+D
*                ---
*
TCP_ETAT         EQU       08H     * rangement de ME1, ME2, MDC
TCP_Ai           EQU       04H     * rangement de A1 et A2
TCP_L            EQU       02H     * rangement de L
TCP_DONNEES      EQU       01H     * rangement des données
*
*
*                XOR
*                ---
XOR_IMMEDIAT     EQU       08H     * opérande "valeur immédiate"
XOR_REGISTRE     EQU       00H     * opérande "registre"

                     * ------------------------------ *
                     * fin du fichier des constantes  *
                     * ------------------------------ *
```

ANNEXE 2 - MESSAGES D'ERREURS ET D'INFORMATIONS

Cette annexe donne la correspondance entre le n° d'un message et son libellé et fournit des explications complémentaires sur sa signification.

1 - INSTRUCTION INCONNUE DANS CETTE SECTION

L'instruction est inconnue pour le compilateur. Il peut s'agir d'une erreur de syntaxe ou d'une instruction qui n'est pas placée dans la bonne section (LECTEUR, PROGRAM...).

2 - MODULE OBJET TROP GROS

Un module objet ne peut avoir une taille supérieure à celle de la mémoire du LECAM.

3 - PARAMETRE LONGUEUR ATTENDU

L'instruction en cours de compilation nécessite une opérande "longueur".

4 - LONGUEUR TROP GRANDE

L'opérande "longueur" d'une instruction est plus grande que celle acceptée par l'instruction.

5 - SEPARATEUR ATTENDU

Le séparateur utilisé entre deux opérandes est la virgule.

6 - DEBORDEMENT REGISTRE (R00 à RFF AUTORISES)

Les valeurs de n° de registres doivent être comprises entre 0 et 255 (00H à FFH).

7 - LONGUEUR INCORRECTE OU ABSENTE

8 - REGISTRE R00 à RFF ATTENDU

9 - NUMERO DE REGISTRE INCORRECT

10 - POINTEUR TABLE DE PROFIL INCORRECT

11 - NOMBRE D'OCTETS TROP GRAND (SUPERIEUR A 32)

12 - LABEL DEJA DEFINIE

13 - TABLE DES SYMBOLES SATUREE (1)

14 - TABLE DES SYMBOLES SATUREE (2)

15 - FICHIER INEXISTANT

16 - MEMOIRE INSUFFISANTE POUR CHARGER LE FICHIER

19 - INSTRUCTION NON TRAITEE

24 - ORDRE CARTE ENTRANT OU SORTANT ATTENDU

25 - LABEL NON DEFINI

26 - 'ECHO' ou 'NOECHO' ATTENDU

27 - LABEL ATTENDUE

28 - LONGUEUR NULLE

29 - PARAMETRE TROP LONG

30 - PARAMETRE ORIGINE ABSENT (PROGRAM <adresse>)

31 - ADRESSE PROGRAMME INCORRECTE

32 - ADRESSE PROGRAMME SUPERIEURE A MEMOIRE LECAM

33 - RECOUVREMENT AVEC PROGRAMME DEJA COMPILE

34 - 'END' ou 'PROGRAM' ATTENDU

35 - FIN DE FICHIER ATTENDUE

36 - VALEUR IMMEDIATE ATTENDUE

37 - VALEUR SUPERIEURE A 255 (#FF)

38 - VALEUR SUPERIEURE A 65535 (#FFFF)

39 - COMPTE ATTENDU

40 - COMPTE TROP GRAND (SUPERIEUR A 8)

41 - PARAMETRE DE TCP INCORRECT

42 - PARAMETRE DE TCP ABSENT OU NUL

43 - BIT ATTENDU

44 - VALEUR HORS BORNES

45 - MODE DE LA COMMANDE ABORT ATTENDU

47 - DEFINITION D'UNE EQUIVALENCE SANS NOM

48 - ERREUR DANS LA DEFINITION DES LABELS, VOIR LA MAP

58 - NOM DE FICHIER .OBJ et .SYS ATTENDU

59 - PROGRAMME SANS NOM

60 - TROP DE PROGRAMES, TABLE DES SYMBOLES SATUREE

61 - ENDP : NOM DE PROGRAMME INCORRECT OU MANQUANT

62 - FIN DE FICHIER INNATENDUE

63 - PROGRAMME NON DEFINI

64 - FICHIER INEXISTANT

66 - PAS DE FICHIER .OBJ ET .SYS

68 - PAS DE PROGRAMME DEFINI

71 - OPERANDE INCORRECTE

72 - CHAINE DE LA COMMANDE DISP INCOMPLETE

73 - NOM DE L'ETIQUETTE INCORRECTE (REGISTRE)

74 - ERREUR : FICHIER INCOMPLET (.LST, .OBJ ou .SYM)

76 - ADRESSE DE DEBUT SUPERIEURE ADRESSE DE FIN

77 - 'PROGRAM' ATTENDU

159 - INCLUSION INTERDITE DANS UN FICHIER INCLU

161 - BYTE (:B) ou WORD (:W) ATTENDU

ANNEXE 3 - STRUCTURE DES FICHIERS .OBJ ET .SYS

Cette annexe fournit la structure des fichiers .OBJ et .SYS. pour la version 1 de CPLECAM. Le fichier .OBJ contient le code objet résultant de la compilation d'un programme source LECAM (.LEC) par le compilateur. Le .SYS contient le nom des programmes du point .OBJ avec leur adresse de début et de fin.

STRUCTURE DU .OBJ

Le fichier .OBJ à une taille fixe de 1892 octets correspondant en fait à la taille maximum d'un programme pour le LECAM 2. En pratique, ce fichier est l'image mémoire de celle du LECAM. Si un programme doit être téléchargé à l'adresse 0280H et se termine à l'adresse 0300H, il se débutera au déplacement 0280H du fichier .OBJ et se terminera à l'adresse 0300H de ce même fichier.

La déclaration de l'enregistrement correspondant en PASCAL pourrait être :

```
type

MEM_LECAM = record
BUFFER = array[0..1891] of byte;
end;
```

et en C :

***** à compléter *****

STRUCTURE DU .SYS

Le fichier .SYM contient une suite d'enregistrement ayant la structure suivante :

-**nom du programme** : 1er octet = nombre de caractères du nom du programme. Cette longueur ne peut être supérieure à 8. Le premier enregistrement dont le nombre de caractères du nom du fichier vaut 0 (zéro) indique la fin du fichier.
Les octets 2 à 9 contiennent le nom du programme, le premier caractère débutant à l'octet n°2.

-**adresse début** : 2 octets contenant l'adresse de début d'implantation en mémoire du programme. Le premier octet contient les poids forts de l'adresse, le deuxième octet les poids faibles.

-**adresse fin** : 2 octets contenant l'adresse de fin d'implantation du dernier octet du programme. Le premier octet contient les poids forts de l'adresse, le deuxième octet les poids faibles.

Les noms de programmes sont stockés dans l'ordre d'apparition dans le programme source.

La déclaration PASCAL correspondante pourrait être :

ce n'est pas le cas actuellement !!!!!!!!

```
ENR_PROGRAMMES = record
NOM : string[8];
ADDRESSE_DEBUT : word;
ADRESSE_FIN : word;
end;
```

ANNEXE 4 - DIMENSIONNEMENT DE CPLECAM

Cette annexe fournit des indications concernant le dimensionnement de certaines caractéristiques de CPLECAM :

Editeur

-taille maximum d'un fichier source : la seule limitation est la mémoire restant disponible une fois l'éditeur chargé à l'intérieur des 640 ko autorisé par MS-DOS. Le programme ne gère pas de mémoire étendue.

Compilateur

-nombre maximum de programmes pouvant être déclarés dans une unité de compilation : 200.

-nombre maximum de labels pouvant être définis : 1000.

-nombre maximum de labels pouvant être référencés : 1000.

-taille maximum d'un programme objet : 1982 octets (LECAM 2).