Bibliothèque assembleur pour Alcyane
Version 1.1

Table des matières

Révisions

Le tableau suivant présente les révisions du présent document et les améliorations de la bibliothèque assembleur pour Alcyane.

Version	Modifications
1.0	Version initiale : bibliothèques clavier, écran et disquette
1.1	Version contenant les bibliothèques graphique, mathématique, imprimante et V24

Pourquoi une bibliothèque assembleur pour Alcyane

Après avoir publié un émulateur pour Alcyane puis un document décrivant le fonctionnement interne du A6E, il m'a semblé intéressant de publier des bibliothèques en assembleur permettant aux développeurs de mettre en œuvre rapidement et simplement les machines A10, A6 et A6E, au moins pour les fonctions essentielles.

Au début, j'ai tenté d'utiliser l'assembleur original d'Alcyane mais il était vraiment trop limité et ne fonctionnait qu'avec une Alcyane dotée de 56 ou 64 k-octets de RAM ou bien avec l'émulateur. L'outil d'édition intégré était vraiment très pauvre (éditeur ligne par ligne), la capacité du disque était limitée et son fonctionnement était un peu cryptique pour les débutants et même pour les développeurs chevronnés. C'est la raison pour laquelle j'ai décidé d'écrire un macro-assembleur pour Windows ciblant l'Alcyane mais utilisable sur beaucoup d'autres machines. Ce macro-assembleur fait l'objet d'un document séparé.

Le présent document liste les fonctions de la bibliothèque par thèmes : clavier, écran, disque, graphique, calcul binaire, imprimante. Ces bibliothèques sont aisément modifiables et contiennent de nombreux commentaires.

Rassurez-vous, l'assembleur du 8085 est bien plus simple qu'il y paraît ! De plus, de très nombreux exemples sont disponibles sur Internet.

Quoi qu'il en soit, je serai heureux de vous aider à construire quelque chose avec les bibliothèques décrites ci-dessous.

Bon courage !

Michel LUCAS

Principes de base de la bibliothèque

La bibliothèque assembleur pour Alcyane utilise trois principes simples :

- J'ai tenté d'écrire du code lisible, pas toujours optimisé mais aussi simple que possible. Rien n'empêche d'utiliser des parties de la bibliothèque pour développer des logiciels et, pourquoi pas, sur d'autres machines que l'Alcyane.

- À de rares exceptions près, une erreur lors d'un appel à la bibliothèque est toujours signalée par le flag CY à 1 et, la plupart du temps, par un code d'erreur dans le registre A. Les codes d'erreur retournés par la bibliothèque sont ceux du basic Alcyane.

- Pour beaucoup de fonctions de la bibliothèque, il est possible de définir des « fonctions de rappel » (*callback*) qui sont appelées lors de la survenue d'un événement. Par exemple, la fonction `CATAL`, qui permet de parcourir le catalogue d'un disque, possède trois fonctions de rappel : quand l'en-tête du disque est lu, pour chaque fichier trouvé et en fin de catalogue. Ceci permet aussi bien d'afficher le catalogue que de rechercher un fichier selon les fonctions de rappel associées. Le code est ainsi plus simple, moins encombrant et plus modulaire.

Bibliothèque clavier

Cette bibliothèque cible le clavier à transmission série des A6 et A6E. Je ne possède aucune information sur les claviers à liaison parallèle des A10 mais si je récupère des binaires pour A10, j'étendrai cette partie.

La bibliothèque du clavier ne contient que deux fichiers :

- `clavier.asm`
- `carspec.inc`

Le fichier carspec.inc

Le fichier `carspec.inc` contient les définitions des codes des touches spéciales du clavier Alcyane comme RUN, CONTINUE, CLEAR, STOP et les touches du pavé numérique. Il s'agit d'un fichier destiné à être inclus dans tout fichier source utilisant ces codes de touches. Son utilisation améliore la lisibilité du code.

Le fichier clavier.asm

Le fichier `clavier.asm` contient toutes les définitions et fonctions nécessaires pour faire fonctionner le clavier à transmission série des A6 et A6E.

Les fonctions de la bibliothèque du clavier

Les fonctions de la bibliothèque sont listées dans le tableau suivant :

Nom	Utilisation
`RAZCLAV`	Réinitialisation du clavier en mode basic, traitement de texte ou APL
`CLAAPPUI`	Détection d'appui sur une touche du clavier
`CLAVLECT`	Attente d'appui sur une touche du clavier
`CLAVCLI`	Attente d'appui sur une touche du clavier avec curseur clignotant
`CLAVLECC`	Lecture d'un caractère avec curseur clignotant
`CLAVLBUFF`	Lecture d'un buffer avec gestion des touches →, ←, CLEAR, INST et ENTER

RAZCLAV

Fonction

Réinitialisation du clavier en mode basic, traitement de texte ou APL

En entrée

A = 00H pour le mode basic
A = 01H pour le mode traitement de textes
A = 02H pour le mode APL

En sortie

A est modifié
CY = 0 si l'initialisation a eu lieu
CY = 1 en cas d'erreur (mode incorrect, par exemple)
`ETATCLAV` est initialisé (aucune touche shift / typ / ctrl enfoncée)
`INSRFP` est initialisé en mode insertion
`VITCLIGN` est initialisé (vitesse de clignotement moyenne)

Exemple

```
XRA  A         ;A = 0 -> mode basic
CALL RAZCLAV   ;RAZ du clavier
JC   ERREUR    ;saut si erreur
```

CLAAPPUI

Fonction

Détermine si une touche a été enfoncée

En entrée

Rien

En sortie

CY = 1 si une touche a été enfoncée
CY = 0 si aucune touche n'a été enfoncée
A est modifié

Exemple

```
BOUCLE CALL CLAAPPUI    ;test si touche enfoncée
       JNC  BOUCLE      ;saut si pas de touche enfoncée
```

CLAVLECT

Fonction

Attente d'appui sur une touche, renvoi du code de la touche
Note : les codes des touches pour les différents modes de fonctionnement du clavier sont listés dans le document « Alcyane vu de l'intérieur »

En entrée

Rien

En sortie

CY = 0
A contient le code de la touche frappée
`ETATCLAV` est mis à jour en fonction des touches shift, typ et ctrl

Exemple

```
BOUCLE CALL CLAVLECT    ;attente de pression d'une touche
       CPI  CAR_ENTER   ;test si touche Entrée
       JNZ  BOUCLE      ;saut si pas touche Entrée
```

CLAVCLI

Fonction

Définition de la vitesse de clignotement du curseur

En entrée

A = 00H .. 07H (00H = curseur fixe, 01H = clignotement lent, 07H = clignotement rapide)

En sortie

CY = 0 en l'absence d'erreur
CY = 1 si A > 07H en entrée

Exemple

```
MVI  A,05H     ;vitesse de clignotement moyenne
CALL CLAVCLI   ;définition de la vitesse de clignotement
```

CLAVLECC

Fonction

Attente de frappe au clavier avec curseur clignotant

En entrée

HL = pointeur sur l'emplacement du curseur à l'écran

En sortie

A = code de la touche frappée
`ETATCLAV` est mis à jour (touches shift, typ et ctrl)

Exemple

```
LXI  H,1080H   ;ligne 2, colonne 1
CALL CLAVLECC  ;attente de frappe
```

CLAVLBUF

Fonction

Affichage d'un buffer prérempli et édition à l'écran avec gestion du curseur
Gestion de la touche INST en bascule insertion / refrappe
Gestion de la touche DEL (suppression du caractère sous le curseur)
Gestion des touches curseur droite et gauche
Gestion de la touche CLEAR (effacement total)
Sortie par la frappe de la touche Entrée

En entrée

HL = pointeur sur le buffer prérempli qui doit se terminer par un 0DH
BC = longueur maximale du buffer
`PTRAFF` = adresse écran de début d'affichage du buffer

En sortie

CY = 1 si débordement du buffer en entrée ou lors de la frappe
CY = 0 en fin d'édition (frappe de le touche Entrée)
A est modifié

Exemple

```
          LXI  H,1180H           ;ligne 4, colonne 1
          SHLD PTRAFF
          LXI  H,BUFCLAV         ;HL pointe le buffer clavier
          LXI  B,SIZE (BUFCLAV)  ;BC = taille maxi
          CALL CLAVLBUF
          RET

BUFCLAV DB 'Test de buffer clavier        ', 0DH
```

Note

Si l'adresse d'affichage à l'écran est trop haute pour l'affichage complet du buffer ou si les caractères saisis nécessitent un défilement de l'écran vers le haut, celui-ci a lieu automatiquement.

Les paramètres et indicateurs de la bibliothèque du clavier

Les paramètres et indicateurs de la bibliothèque du clavier sont listés dans le tableau suivant :

Nom	Taille	Utilisation
`ETATCLAV`	1 octet	État des touches typ, shift, ctrl et pavé numérique. D0 = 1 si shift est enfoncé. D1 = 1 si ctrl est enfoncé. D2 = 1 si typ est enfoncé. D3 = 1 si une touche du pavé numérique est enfoncée.
`INSRFP`	1 octet	Indicateur d'insertion / refrappe. `INSRFP` = 0 en mode insertion, sinon mode refrappe.
`ADRRAPIN`	2 octets	Adresse de la fonction de rappel appelée lors d'un appui sur INST dans `CLAVLBUF`. Normalement, la fonction déjà présente permet de changer automatiquement la valeur de `INSRFP` mais elle peut être surchargée. Si aucune fonction de rappel n'est utilisée, `ADRRAPIN` vaut 0000H.

Bibliothèque écran

Cette bibliothèque cible les cartes affichant 24 lignes de 80 ou 128 caractères des machines A10, A6 et A6E. Le mode 128 colonnes n'est cependant disponibles que sur les cartes d'affichage les plus récentes.

Les fonctions de la bibliothèque de l'écran

Toutes les fonctions de la bibliothèque se trouvent dans le fichier `ecran.asm`. Les fonctions de la bibliothèque sont listées dans le tableau suivant :

Nom	Utilisation
ECRINIT	Initialisation de l'écran en mode 80 ou 128 colonnes et effacement
ECREFTOU	Effacement complet de l'écran
ECREFFBC	Effacement de plusieurs lignes de l'écran
ECRGCAR	Choix du générateur de caractères basic ou APL
TESTNL	Vérification de validité du numéro de ligne (dans l'intervalle 1..24)
ECRLIGN	Définition de la ligne courante d'affichage (dans l'intervalle 1..24)
ECRCOL	Définition de la colonne courante d'affichage (dans l'intervalle 1..80 ou 1..128)
ECRLICO	Définition simultanée de la ligne et de la colonne d'affichage
ECRAFFPM	Affichage d'un caractère sans déplacement du pointeur d'affichage
ECRAFFMC	Affichage d'un caractère avec avance du pointeur d'affichage PTRAFF
ECRLISUI	Passage à la ligne avec gestion du défilement de l'écran vers le haut
ECRAVCUR	Avance du curseur avec gestion du défilement de l'écran vers le haut
ECRRECUR	Recul du curseur avec gestion du défilement de l'écran vers le bas
DEFILHT	Défilement de l'écran d'une ligne vers le haut
DEFILBA	Défilement de l'écran d'une ligne vers le bas
EFFL24	Effacement de la ligne 24 sans défilement de l'écran
EFFL01	Effacement de la ligne 1 sans défilement de l'écran
ADRDEBL	Calcul de l'adresse mémoire de début d'une ligne
ADRFINL	Calcul de l'adresse mémoire de fin d'une ligne
DEFHTBC	Défilement d'une portion d'écran vers le haut
DEFBABC	Défilement d'une portion d'écran vers le bas
ECRAFFCR	Affichage d'une chaîne de caractères terminée par 0DH
ECRAFFLG	Affichage d'une chaîne de caractères sur une longueur donnée
EFFFINLI	Effacement de la fin de la ligne courante
ECRNCAR	Affichage de n caractères identiques

ECRINIT

Fonction

Initialisation du mode d'affichage 80 ou 128 colonnes et effacement de l'écran

En entrée

A = 00H pour le mode 80 colonnes
A différent de 00H pour le mode 128 colonnes

En sortie

A est modifié
`MODEECR` est mis à jour
`PTRAFF` est mis à jour au début de l'écran
`FINPAGE` est mis à jour à la fin de l'écran (en bas à droite)
`MODEINV` est mis à jour (pas d'inversion vidéo)
`DEFILFIN` est mis à jour (défilement lors de l'affichage en bas à droite de l'écran)
`ADRRAPDE` est initialisé à 0000H (pas de rappel sur défilement de l'écran)
L'écran est effacé

Exemple

```
XRA  A          ;mode 80 colonnes
CALL ECRINIT    ;effacement de l'écran
```

Note

Si les cartes présentes dans l'Alcyane ne permettent pas l'affichage en mode 128 colonnes, l'affichage demeure en mode 80 colonnes. L'affichage peut cependant être déformé si le logiciel suppose que 128 colonnes sont affichées.

ECREFTOU

Fonction

Effacement complet de l'écran

En entrée

Rien

En sortie

A est modifié
L'écran est effacé intégralement quel que soit le mode d'affichage

Exemple

```
CALL ECREFTOU    ;effacement complet de l'écran
```

ECREFFBC

Fonction

Effacement d'une portion d'écran (lignes complètes)

En entrée

B = numéro de la première ligne à effacer (1..24)
C = numéro de la dernière ligne à effacer (1..24)
C doit être supérieur ou égal à B

En sortie

A est modifié
CY = 0 si l'effacement a eu lieu
CY = 1 si erreur (B ou C invalide, B < C)

Exemple

```
MVI  B,16N      ;première ligne effacée = 16
MVI  C,21N      ;dernière ligne effacée = 21
CALL  ECREFBC   ;effacement des lignes 16 à 21 incluses
```

ECRGCAR

Fonction

Choix du générateur de caractères d'affichage (basic ou APL)

En entrée

A = 00H pour les caractères du basic
A différent de 00H pour les caractères APL

En sortie

A est modifié
`MODEECR` est mis à jour

Exemple

```
XRA  A          ;générateur de caractères basic
CALL ECRGCAR    ;sélection du générateur de caractères
```

Note

Cette fonction nécessite une ROM générateur de caractères qui contient les jeux de caractères basic et APL. Si le second jeu est absent, l'écran restera noir.

ECRRAZIV

Fonction

Arrêt du mode inversion vidéo

En entrée

Rien

En sortie

Les registres sont inchangés

Exemple

```
CALL ECRRAZIV    ;arrêt du mode inversion vidéo
```

ECRRESOL

Fonction

Choix de la résolution de l'écran (80 ou 128 colonnes)

En entrée

A = 00H pour 80 colonnes
A <> 00H pour 128 colonnes

En sortie

A est modifié
`MODEECR` est mis à jour
`FINPAGE` est mis à jour

Exemple

```
MVI  A,01H      ;mode 128 colonnes demandé
CALL ECRRESOL   ;passage en mode 128 colonnes
```

Note

L'appel de cette fonction ne modifie pas le contenu de la mémoire écran ni le générateur de caractères sélectionné.

TESTNL

Fonction

Vérifie si le numéro de ligne indiqué est valide

En entrée

A = numéro de ligne (1..24)

En sortie

CY = 0 si le numéro de ligne est compris entre 1 et 24
CY = 1 si le numéro de ligne est invalide

Exemple

```
MVI  A,35N    ;ligne 35
CALL TESTNL   ;test de validité
JC   ERREUR   ;saut si erreur
```

ECRLIGN

Fonction

Positionne le point d'affichage sur la ligne désirée quel que soit le mode d'affichage (80 ou 128 colonnes)

En entrée

A = numéro de ligne (1..24)

En sortie

CY = 0 si le numéro de ligne est compris entre 1 et 24
CY = 1 si le numéro de ligne est invalide
`PTRAFF` est mis à jour et pointe au début de la ligne demandée

Exemple

```
MVI  A,12N     ;ligne 12
CALL ECRLIGN   ;PTRAFF vaut 1580H
```

ECRCOL

Fonction

Positionne le point d'affichage sur la colonne désirée

En entrée

A = numéro de colonne (1..80 ou 1..128 selon le mode d'affichage courant)

En sortie

CY = 0 si le numéro de colonne est valide
CY = 1 si le numéro de colonne est invalide (> 80 ou > 128 selon le mode d'affichage)
`PTRAFF` est mis à jour et pointe la colonne demandée sur la ligne courante (voir `ECRLIGN`)

Exemple

```
        MVI  A,01H
        CALL ECRINIT  ;mode 128 colonnes
        MVI  A,12N
        CALL ECRLIGN  ;ligne 12
        MVI  A,(128-SIZE (TEXTE)) / 2
        CALL ECRCOL   ;colonne pour texte centré
        LXI  H,TEXTE
        CALL ECRAFFCR ;affiche le texte centré
        RET
TEXTE DB 'Titre de la page au milieu de l''écran', 0DH
```

ECRLICO

Fonction

Positionne le point d'affichage sur la ligne et la colonne désirées

En entrée

B = numéro de ligne (1..24)
C = numéro de colonne (1..80 ou 1..128 selon le mode d'affichage)

En sortie

CY = 0 si ligne et colonne sont valides
CY = 1 si ligne ou colonne est invalide
`PTRAFF` est mis à jour, il est invalide si CY = 1

Exemple

```
LXI  B,0410H   ;ligne 4, colonne 16
CALL ECRLICO   ;positionne le point d'affichage
```

ECRAFFPM

Fonction

Affichage d'un caractère au point d'affichage sans modifier `PTRAFF`

En entrée

A = code ASCII du caractère à afficher
`PTRAFF` est le point d'affichage du caractère

En sortie

Le caractère est affiché

Exemple

```
LXI  B,0A05H    ;ligne 10, colonne 5
CALL ECRLICO    ;positionne PTRAFF
MVI  A,'*'      ;caractère à afficher
CALL ECRAFFPM   ;affichage
```

ECRAFFMC

Fonction

Affichage d'un caractère au point d'affichage et progression de `PTRAFF`

En entrée

A = code ASCII du caractère à afficher
`PTRAFF` est le point d'affichage du caractère

En sortie

Le caractère est affiché et `PTRAFF` est mis à jour.
Si nécessaire, l'écran défile vers le haut

Exemples

```
;affichage sans défilement
MVI  B,12N     ;ligne 12
MVI  C,25N     ;colonne 25
CALL ECRLICO   ;positionne PTRAFF
;affichage de 'ABC' ligne 12, colonnes 25, 26 et 27
MVI  A,'A'     ;caractère à afficher
CALL ECRAFFMC
MVI  A,'B'
CALL ECRAFFMC
MVI  A,'C'
CALL ECRAFFMC

;affichage avec défilement
MVI  B,24N     ;ligne 24
MVI  C,80N     ;colonne 80
CALL ECRLICO
;affichage de '$' avec défilement de l'écran
MVI  A,'$'     ;caractère à afficher
CALL ECRAFFMC  ;affichage avec défilement
```

ECRLISUI

Fonction

Passage à la ligne suivante avec défilement de l'écran si nécessaire

En entrée

`PTRAFF` est le point d'affichage

En sortie

`PTRAFF` est mis à jour
L'écran défile si nécessaire

Exemple

```
        ;passage à la ligne sans défilement de l'écran
        MVI  B,12N
        MVI  A,01N
        CALL ECRLICO    ;ligne 12, colonne 1
        LXI  H,TEXTE1
        CALL ECRAFFCR
        CALL ECRLISUI   ;PTRAFF pointe le début de la ligne 13

        ;passage à la ligne avec défilement de l'écran
        MVI  B,24N
        MVI  A,01N
        CALL ECRLICO    ;ligne 24, colonne 1
        LXI  H,TEXTE1
        CALL ECRAFFCR
        CALL ECRLISUI   ;PTRAFF pointe le début de la ligne 24
                        ;un défilement de l'écran a eu lieu

TEXTE1 DB  'Test'
```

ECRAVCUR

Fonction

Avance le point d'affichage d'un caractère avec défilement de l'écran si nécessaire
Fonctionne en mode 80 ou 128 colonnes

En entrée

`PTRAFF` est le point d'affichage courant

En sortie

`PTRAFF` est mis à jour
L'écran défile vers le haut si nécessaire

Exemple

```
MVI  A,'*'
CALL ECRAFFPM   ;affiche une '*' au point d'affichage courant
CALL ECRAVCUR   ;avance le point d'affichage de 2 positions
CALL ECRAVCUR
```

ECRRECUR

Fonction

Recule le point d'affichage d'un caractère avec défilement de l'écran si nécessaire
Fonctionne en mode 80 ou 128 colonnes

En entrée

`PTRAFF` est le point d'affichage courant

En sortie

`PTRAFF` est mis à jour
L'écran défile vers le bas si nécessaire

Exemple

```
MVI  A,'*'
CALL ECRAFFMC   ;affiche une '*' et avance le point d'affichage
CALL ECRRECUR   ;recule le point d'affichage
```

DEFILHT

Fonction

Défilement de l'écran vers le haut en mode 80 ou 128 colonnes
La ligne 24 est effacée

En entrée

Rien

En sortie

A est modifié
`PTRAFF` est mis à jour au début de la ligne 24

Exemple

```
CALL DEFILHT
```

DEFILBA

Fonction

Défilement de l'écran vers le bas en mode 80 ou 128 colonnes
La ligne 1 est effacée

En entrée

Rien

En sortie

A est modifié
`PTRAFF` est mis à jour au début de la ligne 1

Exemple

```
CALL DEFILBA
```

EFFL24

Fonction

Effacement de la ligne 24

En entrée

Rien

En sortie

Tous les registres sont sauvegardés

Exemple

```
CALL EFFL24
```

EFFL01

Fonction

Effacement de la ligne 1

En entrée

Rien

En sortie

Tous les registres sont sauvegardés

Exemple

```
CALL EFFL01
```

ADRDEBL

Fonction

Calcul de l'adresse mémoire de début d'une ligne

En entrée

A = numéro de la ligne concernée (1..24)

En sortie

HL = adresse de début de la ligne concernée

Exemple

```
MVI  A,07H      ;ligne 7
CALL ADRDEBL    ;HL = 1300H
```

Note

Pour des raisons d'optimisation, il n'y a aucun contrôle de validité du numéro de ligne

ADRFINL

Fonction

Calcul de l'adresse mémoire de début d'une ligne

En entrée

A = numéro de la ligne concernée (1..24)

En sortie

HL = adresse de fin de la ligne concernée

Exemple

```
XRA  A
CALL ECRRESOL   ;mode 80 colonnes
MVI  A,07H      ;ligne 7
CALL ADRFINL    ;HL = 134FH
```

Note

Pour des raisons d'optimisation, il n'y a aucun contrôle de validité du numéro de ligne

DEFHTBC

Fonction

Défilement vers le haut d'une portion d'écran

En entrée

B = numéro de la première ligne du bloc à faire défiler vers le haut
C = numéro de la dernière ligne du bloc à faire défiler vers le haut
C doit être supérieur à B

En sortie

CY = 0 si les numéros de lignes sont corrects
CY = 1 s'il y a une erreur dans les numéros de ligne, aucun défilement dans ce cas
La première ligne (numéro donné par B) disparaît
La dernière ligne (numéro donné par C) est effacée

Exemple

```
MVI  B,07N    ;début = ligne 7
MVI  C,12N    ;fin = ligne 12
CALL DEFHTBC  ;défilement vers le haut des lignes 7 à 12
```

DEFBABC

Défilement vers le bas d'une portion d'écran

En entrée

B = numéro de la première ligne du bloc à faire défiler vers le bas
C = numéro de la dernière ligne du bloc à faire défiler vers le bas
C doit être supérieur à B

En sortie

CY = 0 si les numéros de lignes sont corrects
CY = 1 s'il y a une erreur dans les numéros de ligne, aucun défilement dans ce cas
La dernière ligne (numéro donné par B) disparaît
La première ligne (numéro donné par C) est effacée

Exemple

```
MVI  B,08N    ;début = ligne 8
MVI  C,13N    ;fin = ligne 13
CALL DEFBABC  ;défilement vers le bas des lignes 8 à 13
```

ECRAFFCR

Fonction

Affichage de texte terminé par CR (0DH) avec gestion du code d'inversion vidéo
Défilement de l'écran si nécessaire
Fonction de rappel définissable en cas de défilement de l'écran

En entrée

HL = adresse de la chaîne de caractères à afficher
`PTRAFF` = point d'affichage du texte
`CARINVER` = code ASCII du caractère permettant l'inversion vidéo (0EH par défaut)
`DEFILFIN` = indicateur de défilement en fin d'écran (00H = pas de défilement, autre valeur = défilement)
`ADRRAPDE` = pointeur sur la fonction appelée en cas de défilement d'écran (0000H si indéfinie).

En sortie

A est modifié
`PTRAFF` est mis à jour
L'écran a défilé vers le haut si nécessaire

Exemple

```
      ;affichage simple avec inversion vidéo
      LXI  B,0501H     ;ligne 5, colonne 1
      CALL ECRLICO     ;définit le point d'affichage
      CALL ECRRAZIV    ;pas d'inversion vidéo
      LXI  H,TEXTE     ;HL pointe le texte à afficher
      CALL ECRAFFCR    ;affiche le texte
      RET
TEXTE DB 'Normal ', 0EH, 'Inversé', 0EH, ' Normal', 0DH
```

Notes

L'utilisation de `DEFILFIN` permet d'afficher toute la ligne 24 sans provoquer de défilement d'écran
Le mode d'inversion vidéo est conservé en sortie de la fonction `ECRAFFCR`

ECRAFFLG

Fonction

Affichage de texte d'une longueur déterminée avec gestion du code d'inversion vidéo
Gestion du passage à la ligne par le caractère CR (0DH)
Défilement de l'écran si nécessaire
Fonction de rappel définissable en cas de défilement de l'écran

En entrée

HL = adresse de la chaîne de caractères à afficher
BC = nombre de caractères à afficher
`PTRAFF` = point d'affichage du texte
`CARINVER` = code ASCII du caractère permettant l'inversion vidéo (0EH par défaut)
`DEFILFIN` = indicateur de défilement en fin d'écran (00H = pas de défilement, autre valeur = défilement)
`ADRRAPDE` = pointeur sur la fonction appelée en cas de défilement d'écran (0000H si indéfinie).

En sortie

A est modifié
`PTRAFF` est mis à jour
L'écran a défilé vers le haut si nécessaire

Exemple

```
      ;affichage sur 3 lignes, inversion vidéo de la ligne 2
      LXI  B,0501H     ;ligne 5, colonne 1
      CALL ECRLICO     ;définit le point d'affichage
      CALL ECRRAZIV    ;pas d'inversion vidéo
      LXI  H,TEXTE     ;HL pointe le texte à afficher
      LXI  B,SIZE (TEXTE)
      CALL ECRAFFLG    ;affiche le texte
      RET
TEXTE DB 'Normal', 0DH, 0EH, 'Inversé', 0EH, 0DH, 'Normal'
```

Notes

L'utilisation de `DEFILFIN` permet d'afficher toute la ligne 24 sans provoquer de défilement d'écran
Le mode d'inversion vidéo est conservé en sortie de la fonction `ECRAFFCR`

EFFFINLI

Fonction

Effacement de la fin de la ligne courante à partir de `PTRAFF`

En entrée

Rien

En sortie

`PTRAFF` est mis à jour

Exemple

```
        LXI  B,0601H   ;ligne 6, colonne 1
        CALL ECRLICO   ;définit le point d'affichage
        LXI  H,TEXTE
        CALL ECRAFFCR  ;affiche le texte
        CALL EFFFINLI  ;efface la fin de la ligne
        RET
TEXTE DB  'Test d''effacement de fin de ligne', 0DH
```

ECRNCAR

Fonction

Affichage de plusieurs caractères identiques
Gestion du passage à la ligne en mode 80 ou 128 colonnes
Pas de défilement de l'écran

En entrée

A = code ASCII du caractère à afficher
BC = nombre de caractères à afficher
HL = adresse du point d'affichage sur l'écran

En sortie

A est modifié
Le texte est affiché

Exemple

```
MVI  A,09H    ;semi-graphique trait horizontal
LXI  B,33N    ;33 répétitions
CALL ECRNCAR  ;affichage
```

Note

Cette fonction est surtout utile pour tracer des tableaux

Les paramètres et indicateurs de la bibliothèque de l'écran

Les paramètres et indicateurs de la bibliothèque de l'écran sont listés dans le tableau suivant :

Nom	Taille	Utilisation
PTRAFF	2 octets	Pointeur d'affichage courant
FINPAGE	2 octets	Pointeur sur la dernière case mémoire de l'écran
MODEECR	1 octet	Mode courant d'affichage D7 = 0 en mode 80 colonnes, D7 = 1 en mode 128 colonnes D0 = 0 pour le générateur de caractères basic, D0 = 1 pour les caractères APL Les bits D1 à D6 sont inutilisés Utilisez ECRINIT, ECRRESOL et/ou ECRGCAR pour modifier le mode d'affichage
CARINVER	1 octet	Caractère utilisé comme balise pour l'inversion vidéo Par défaut, ce caractère vaut 0EH Il peut être modifié au besoin
DEFILFIN	1 octet	Indicateur de demande de défilement d'écran lors de l'affichage du caractère situé en bas à droite. DEFILFIN = 0, pas de défilement de l'écran DEFILFIN <> 0, défilement en fin d'écran
ADRRAPDE	2 octets	Pointeur sur le sous-programme appelé en cas de défilement de l'écran lors de l'appel à ECRAFFCR et ECRAFFLG Par défaut, ce pointeur vaut 0000H Le sous-programme appelé doit sauvegarder tous les registres

Bibliothèque disquette

Cette bibliothèque cible les lecteurs de disquettes double densité 3,5 pouces, 5,25 pouces et 8 pouces interfacés par l'ensemble de cartes 3010C-05 et 9040.

La bibliothèque de gestion des disquettes contient de nombreux fichiers :

- `EMI.asm`
- `INDATA.asm`
- `INM.asm`
- `INMES.asm`
- `disque.asm`
- `catal.asm`
- `ecrific.asm`
- `lecrecfmt.asm`
- `lirefic.asm`
- `rechfic.asm`
- `suppfic.asm`

Les fichiers EMI.asm, INDATA.asm, INM.asm et INMES.asm

Ces quatre fichiers contiennent les sous-programmes de bas niveau permettant de dialoguer avec l'interface 3010C-05. Il est possible de les consulter pour en comprendre le fonctionnement mais je déconseille fortement de les modifier, le code étant optimisé et contenant, entre autres, des instructions auto-modifiées.

Le dialogue avec l'interface est exposé en détail au chapitre « L'interface du contrôleur de disquettes » de mon document intitulé « Alcyane A6E vu de l'intérieur ».

Le fichier disque.asm

Ce fichier contient toutes les définitions de structures, tables et variables globales de la bibliothèque. Il définit les sous-ensembles constituant la bibliothèque au moyen de directives `USE`.

Le fichier catal.asm

Ce fichier contient le sous-programme `CATAL` qui permet le parcours du catalogue d'un disque avec des fonctions de rappel définissables pour la lecture de l'en-tête du disque, pour chaque fichier rencontré et à la fin du catalogue.

Le fichier ecrific.asm

Ce fichier contient le sous-programme `ECRIFIC` permettant d'écrire un fichier au format Alcyane sur un disque.

Le fichier lececrfmt.asm

Ce fichier contient les sous-programmes de base de lecture (`DSQLES`) et d'écriture (`DSQECS`) d'un secteur ainsi que de formatage (`DSQFMT`) d'un disque.

Le fichier lirefic.asm

Ce fichier contient le sous-programme `LIREFIC` permettant de lire en mémoire un fichier au format Alcyane.

Le fichier rechfic.asm

Ce fichier contient le sous-programme `RECHFIC` permettant de rechercher un fichier sur un disque. Ce sous-programme est utilisé lors de l'écriture et de la suppression d'un fichier.

Le fichier suppfic.asm

Ce fichier contient le sous-programme `SUPPFIC` permettant de supprimer un fichier Alcyane sur disque.

Les fonctions de la bibliothèque de gestion des disquettes

Les fonctions de la bibliothèque sont listées dans le tableau suivant :

Nom	Utilisation
`CATAL`	Parcours du catalogue avec fonctions de rappel
`ECRIFIC`	Écriture d'un fichier au format Alcyane
`DSQLES`	Lecture d'un secteur d'un disque
`DSQECS`	Écriture d'un secteur d'un disque
`DSQFMT`	Formatage d'un disque
`LIREFIC`	Lecture d'un fichier au format Alcyane en mémoire
`RECHFIC`	Recherche d'un fichier sur un disque
`SUPPFIC`	Suppression d'un fichier

CATAL

Fichier source

`catal.asm`

Fonction

Parcours du catalogue d'un disque avec des fonctions de rappel sur lecture de l'en-tête, sur découverte d'un descripteur de fichier et sur fin de catalogue

En entrée

A = numéro du disque (1, 2, 9, 10, 11 ou 12)
`ADRAPENT` = adresse du sous-programme appelé lors de la lecture de l'en-tête
`ADRAPFIC` = adresse du sous-programme appelé lors de la découverte d'un nom de fichier
`ADRAPFIN` = adresse du sous-programme appelé quand la fin du catalogue est atteinte

En sortie

CY = 0 en l'absence d'erreur
CY = 1 en cas d'erreur (A contient alors le numéro de l'erreur)
`NOSCCAT` = numéro du secteur courant du catalogue utilisable dans les sous-programmes de rappel

Exemple

Un exemple d'utilisation de `CATAL` se trouve dans le sous-programme `RECHFIC`. Il y a de nombreux commentaires permettant d'en cerner le fonctionnement.

ADRAPENT

Fonction

Adresse de la fonction de rappel invoquée lorsque le sous-programme `CATAL` lit l'en-tête d'une disquette

Utilisation

Si ce rappel n'est pas utilisé, il faut initialiser `ADRAPENT` à la valeur 0000H

En entrée

La paire de registres HL pointe sur le premier octet du premier secteur du catalogue

En sortie

CY = 0 si le secteur numéro 1 de la disquette a été lu
CY = 1 si la disquette n'est pas formatée

Note

Le sous-programme appelé doit sauvegarder tous les registres sauf A et les flags

ADRAPFIC

Fonction

Adresse de la fonction de rappel invoquée lorsque le sous-programme `CATAL` a détecté un nom de fichier dans le catalogue

Utilisation

Si ce rappel n'est pas utilisé, il faut initialiser `ADRAPFIC` à la valeur 0000H

En entrée

La paire de registres HL pointe le début d'un descripteur de fichier (structure `STRENT`)
La paire de registres BC contient le numéro de secteur courant du catalogue

En sortie

CY = 0 si la lecture du catalogue doit se poursuivre
CY = 1 si la lecture du catalogue doit être interrompue (fichier trouvé par exemple)

Note

Le sous-programme appelé doit sauvegarder tous les registres sauf A et les flags

ADRAPFIN

Fonction

Adresse de la fonction de rappel invoquée lorsque le sous-programme `CATAL` a détecté la fin du catalogue

Utilisation

Si ce rappel n'est pas utilisé, il faut initialiser `ADRAPFIN` à la valeur 0000H

En entrée

La paire de registres HL pointe sur le FFH de fin de catalogue
La paire de registres DE contient le nombre d'octets restants dans le secteur courant
La paire de registres BC contient le numéro de secteur courant du catalogue

En sortie

CY = 0

Note

Le sous-programme appelé doit sauvegarder tous les registres sauf A et les flags
La valeur des registres DE permet de savoir s'il reste assez de place pour créer un nom dans le secteur courant du catalogue ou s'il faut étendre le catalogue au secteur suivant

ECRIFIC

Fichier source

`ecrific.asm`

Fonction

Écriture d'un fichier au format Alcyane sur une disquette
Si le fichier n'existe pas, il est créé après confirmation

En entrée

A = numéro du disque (1, 2, 9, 10, 11 ou 12)
La paire de registres HL pointe sur le descripteur de fichier (structure `STRDFIC`)
La paire de registres DE pointe sur l'adresse de début de la zone mémoire à écrire
La paire de registres BC contient le nombre d'octets à écrire
`ADRAPEXI` = adresse du sous-programme appelé si le fichier existe déjà

En sortie

CY = 0 si le fichier a été écrit
CY = 1 en cas d'erreur (A contient alors le numéro de l'erreur)

Exemple

Plusieurs exemples d'écriture de fichier se trouvent dans le programme de test de la bibliothèque

Note

L'algorithme d'écriture, assez complexe, est donné dans le fichier `ecrific.asm`

ADRAPEXI

Fonction

Adresse de la fonction de rappel invoquée si le fichier à écrire existe déjà
Ce sous-programme permet de demander l'accord de l'utilisateur avant l'écriture

Utilisation

Si ce rappel n'est pas utilisé, il faut initialiser `ADRAPEXI` à la valeur 0000H

En entrée

Rien

En sortie

CY = 0 si l'écrasement du fichier est autorisé
CY = 1 si l'écriture est refusée

Note

Le sous-programme appelé doit sauvegarder tous les registres sauf A et les flags

DSQLES

Fichier source

`lececrfmt.asm`

Fonction

Lecture d'un secteur de 256 octets sur une disquette

En entrée

A = numéro du disque (1, 2, 9, 10, 11 ou 12)
La paire de registres HL pointe sur l'emplacement mémoire où lire les données
La paire de registres BC contient le numéro du secteur à lire (1..n)
`MONOSCT` indique une lecture mono-secteur (00H) ou multi-secteurs (40H)

En sortie

CY = 0 si le secteur a été lu
CY = 1 en cas d'erreur (A contient alors le numéro de l'erreur)

Exemple

Plusieurs exemples de lecture de secteur se trouvent dans la bibliothèque, notamment dans la fonction `CATAL`

Note

L'interface 3010C-05 optimise les lectures sur disque de façon à éviter de multiples accès au disque pour la lecture de secteurs consécutifs (mode multi-secteurs)

DSQECS

Fichier source

`lececrfmt.asm`

Fonction

Écriture d'un secteur de 256 octets sur une disquette

En entrée

A = numéro du disque (1, 2, 9, 10, 11 ou 12)
La paire de registres HL pointe sur l'emplacement mémoire à écrire sur disque
La paire de registres BC contient le numéro du secteur à écrire (1..n)
`MONOSCT` indique une écriture mono-secteur (00H) ou multi-secteurs (40H)

En sortie

CY = 0 si le secteur a été lu
CY = 1 en cas d'erreur (A contient alors le numéro de l'erreur)

Exemple

Plusieurs exemples d'écriture de secteur se trouvent dans la bibliothèque, notamment dans la fonction `ECRIFIC`

Note

L'interface 3010C-05 optimise les écritures sur disque de façon à éviter de multiples accès au disque pour l'écriture de secteurs consécutifs (mode multi-secteurs)

DSQFMT

Fichier source

`lececrfmt.asm`

Fonction

Formatage de bas niveau d'une disquette

En entrée

A = numéro du disque (1, 2, 9, 10, 11 ou 12)

En sortie

CY = 0 si la disquette est formatée
CY = 1 en cas d'erreur (A contient alors le numéro de l'erreur)

Exemple

Un exemple se trouve dans le programme de test de la bibliothèque

Note

Le formatage de bas niveau n'écrit pas un catalogue « vide » sur la disquette
Le programme de test de la bibliothèque peut être utilisé comme exemple de création d'un catalogue « vide »

LIREFIC

Fichier source

`lirefic.asm`

Fonction

Lecture d'un fichier au format Alcyane

En entrée

A = numéro du disque (1, 2, 9, 10, 11 ou 12)
La paire de registres HL pointe sur le descripteur du fichier à lire (structure `STRDFIC`)
La paire de registres DE pointe la zone mémoire où le fichier sera lu

En sortie

CY = 0 si le fichier a été lu
CY = 1 en cas d'erreur (A contient alors le numéro de l'erreur)

Exemple

Un exemple se trouve dans le programme de test de la bibliothèque

Note

Dans cette version, seuls les fichiers simples peuvent être lus

RECHFIC

Fichier source

`rechfic.asm`

Fonction

Recherche d'un fichier sur disque

En entrée

A = numéro du disque (1, 2, 9, 10, 11 ou 12)
La paire de registres HL pointe sur le nom du fichier recherché (6 caractères)

En sortie

CY = 0 si le fichier a été trouvé
CY = 1 en cas d'erreur (A contient alors le numéro de l'erreur)
`RESURECH` contient le résultat de la recherche :

- `CTROUVE` : le fichier a été trouvé
- `CPASFMT` : le disque n'est pas formaté
- `CPASTRO` : le fichier n'a pas été trouvé

Si le fichier a été trouvé, la paire de registres HL pointe sur le descripteur du fichier (structure `STRDFIC`) et la paire de registres BC contient le numéro de secteur du catalogue où le fichier a été trouvé

Exemple

Un exemple se trouve dans le sous-programme `ECRIFIC`

SUPPFIC

Fichier source

`suppfic.asm`

Fonction

Suppression d'un fichier sur disque

En entrée

A = numéro du disque (1, 2, 9, 10, 11 ou 12)
La paire de registres HL pointe sur le descripteur du fichier recherché (structure `STRDFIC`)

En sortie

CY = 0 si le fichier a été supprimé
CY = 1 en cas d'erreur (A contient alors le numéro de l'erreur)

Exemple

Un exemple se trouve dans le programme de test de la bibliothèque

Les paramètres et indicateurs de la bibliothèque de gestion des disquettes

Les paramètres et indicateurs de la bibliothèque de gestion des disquettes sont listés dans le tableau suivant :

Nom	Taille	Utilisation
`FLGINI`	1 octet	Indique si l'interface a déjà été initialisée (00H) ou pas (55H).
`MONOSCT`	1 octet	Indique que les lectures et écritures s'effectuent en mode mono-secteur (00H) ou multi-secteur (40H)
`BUFDSQ`	256 octets	Buffer de lecture / écriture de secteurs
`BUFCAT`	256 octets	Buffer de lecture / écriture du catalogue
`NOSCCAT`	2 octets	Numéro de secteur courant du catalogue
`TBLMAXSC`	24 octets	Table des nombres de secteurs du catalogue par disque
`TBLMAXSD`	24 octets	Table des nombres de secteurs par disque (dépend de la valeur de `DSQ35` et `DSQ525` par une directive `IF`)
`ADRAPENT`	2 octets	Adresse de rappel sur lecture d'en-tête définie dans `catal.asm`
`ADRAPFIC`	2 octets	Adresse de rappel sur découverte d'un descripteur de fichier définie dans `catal.asm`
`ADRAPFIN`	2 octets	Adresse de rappel sur fin de catalogue définie dans `catal.asm`
`ADRAPEXI`	2 octets	Adresse de rappel définie dans `ecrific.asm` et invoquée quand le fichier recherché est trouvé
`RESURECH`	1 octet	Résultat de la recherche de fichier par `RECHFIC` : • `CTROUVE` = 00H si le fichier a été trouvé • `CPASFMT` = 01H si le disque n'est pas formaté • `CPASTRO` = 02H si le fichier n'a pas été trouvé • `CFCREE` = 03H si le fichier a été créé
`PTDESCR`	2 octets	Pointeur sur le descripteur du fichier trouvé

Le fichier `disque.asm` contient des constantes employées par les sous-programmes de dialogue avec l'interface. Il ne faut pas modifier ces valeurs.

Le fichier `disque.asm` définit les structures d'en-tête de disque (`STRENT`) et de descripteur de fichier (`STRDFIC`). Il est recommandé d'employer ces structures autant que possible.

Bibliothèque graphique

Cette bibliothèque cible l'interface graphique 256 x 256 pixels en 2, 4, 8 ou 16 couleurs des A10, A6 et A6E.

Elle contient les fichiers suivants :

- `graph.asm`
- `cercle.asm`
- `grainit.asm`
- `rempli.asm`
- `texteg.asm`
- `vecteur.asm`
- `pixel.asm`

Le fichier graph.asm

Ce fichier contient la définition de la structure d'un vecteur graphique ainsi que la variable définissant le dessin des vecteurs. Il définit les sous-ensembles constituant la bibliothèque au moyen de directives `USE`.

Le fichier cercle.asm

Ce fichier contient le sous-programme de tracé de cercles (`CERCLE`).

Le fichier grainit.asm

Ce fichier contient le sous-programme permettant de positionner l'ensemble de la page graphique à une couleur (`GRAINIT`).

Le fichier rempli.asm

Ce fichier contient le sous-programme permettant de remplir une surface fermée quelconque avec une couleur (`REMPLI`).

Le fichier texteg.asm

Ce fichier contient le sous-programme permettant d'afficher un texte sur la page graphique (`TEXTEG`). Il contient également toutes les définitions vectorielles des caractères.

Le fichier vecteur.asm

Ce fichier contient un sous-programme de tracé de vecteurs (`VECTEUR`) ainsi que deux sous-programmes optimisés pour le tracé de vecteurs horizontaux (`VECTHO`) et verticaux (`VECTVE`).

Le fichier pixel.asm

Ce fichier contient les deux sous-programmes de lecture et d'écriture de pixel.

Les fonctions de la bibliothèque graphique

Les fonctions de la bibliothèque graphique sont listées dans le tableau suivant :

Nom	Utilisation
GRAINIT	Initialisation de la page graphique
CERCLE	Tracé de cercle
REMPLI	Remplissage de surface fermée
TEXTEG	Affichage de texte sur la page graphique
VECTEUR	Tracé de vecteur
VECTVE	Tracé optimisé de vecteur vertical
VECTHO	Tracé optimisé de vecteur horizontal
LIREPIX	Lecture de la couleur d'un pixel
ECRIPIX	Écriture d'un pixel

GRAINIT

Fonction

Initialisation du fond de la page graphique

En entrée

A = couleur de fond (00H..0FH)

En sortie

A est modifié
L'écran graphique est initialisé

Exemple

```
MVI  A,01H     ;couleur 1
CALL GRAINIT   ;page graphique initialisée
```

Note

Les couleurs supportées par cette fonction dépendent du nombre de plans graphiques installés dans l'Alcyane :

- 1 plan : 2 couleurs
- 2 plans : 4 couleurs
- 3 plans : 8 couleurs
- 4 plans : 16 couleurs

CERCLE

Fonction

Tracé de cercle sur la page graphique par l'algorithme de Horn

En entrée

B = abscisse du centre
C = ordonnée du centre
H = rayon du cercle
A = couleur du tracé (00H..0FH)

En sortie

A est modifié

Exemple

```
MVI   B,127N     ;abscisse du centre
MVI   C,127N     ;ordonnée du centre
MVI   H,40N      ;rayon = 40 pixels
MVI   A,01H      ;couleur 1
CALL CERCLE      ;tracé du cercle
```

Note

Le sous-programme `CERCLE` n'effectue pas de contrôle sur les coordonnées et le rayon. Si un cercle « déborde » de la page graphique, le tracé s'effectue de manière symétrique du côté opposé de l'écran.

REMPLI

Fonction

Remplissage de surface fermée

En entrée

D = abscisse d'un point de la surface à remplir
E = ordonnée d'un point de la surface à remplir
A = couleur de remplissage (00H..0FH)

En sortie

Tous les registres sont modifiés

Exemple

```
MVI   D,45N      ;abscisse du point de départ
MVI   E,110N     ;ordonnée du point de départ
MVI   A,05H      ;couleur de remplissage
CALL REMPLI      ;remplissage de la surface
```

Note

Le sous-programme `REMPLI` utilise intensivement la pile du microprocesseur. Si la surface à remplir est trop complexe, le remplissage est interrompu. Les registres ne sont pas sauvegardés en pile pour laisser plus de place exploitable au sous-programme.

VECTEUR

Fonction

Tracé de vecteur sur la page graphique par l'algorithme de Bresenham

En entrée

B = abscisse du point de départ
C = ordonnée du point de départ
D = abscisse du point final
E = ordonnée du point final
A = couleur de tracé (00H..10H)
`PTSVECT` contient le dessin des points tracés (voir ci-après)

En sortie

A est modifié

Exemple

```
MVI  B,10N     ;abscisse du point de départ
MVI  C,20N     ;ordonnée du point de départ
MVI  D,120N    ;abscisse du point final
MVI  E,200N    ;ordonnée du point final
MVI  A,05H     ;couleur de tracé
CALL VECTEUR   ;tracé du vecteur
```

Note

La couleur spéciale 10H permet de tracer un vecteur en inversant les points déjà présents sur la page graphique.

VECTHO

Fonction

Tracé optimisé de vecteur horizontal

En entrée

B = abscisse du point de départ
C = ordonnée du point de départ
D = abscisse du point final
A = couleur de tracé (00H..10H)
`PTSVECT` contient le dessin des points tracés (voir ci-après)

En sortie

A est modifié

Exemple

```
MVI  B,10N     ;abscisse du point de départ
MVI  C,20N     ;ordonnée du point de départ
MVI  D,200N    ;abscisse du point final
MVI  A,01H     ;couleur de tracé
CALL VECTHO    ;tracé du vecteur horizontal
```

Note

La couleur spéciale 10H permet de tracer un vecteur en inversant les points déjà présents sur la page graphique.

VECTVE

Fonction

Tracé optimisé de vecteur vertical

En entrée

B = abscisse du point de départ
C = ordonnée du point de départ
E = ordonnée du point final
A = couleur de tracé (00H..10H)
`PTSVECT` contient le dessin des points tracés (voir ci-après)

En sortie

A est modifié

Exemple

```
MVI  B,10N     ;abscisse du point de départ
MVI  C,20N     ;ordonnée du point de départ
MVI  E,100N    ;ordonnée du point final
MVI  A,01H     ;couleur de tracé
CALL VECTVE    ;tracé du vecteur vertical
```

Note

La couleur spéciale 10H permet de tracer un vecteur en inversant les points déjà présents sur la page graphique.

TEXTEG

Fonction

Affichage de texte sur la page graphique

En entrée

HL = pointeur sur le texte à afficher terminé par 0DH
D = abscisse d'affichage du texte
E = ordonnée d'affichage du texte
B = échelle horizontale (1..n)
C = échelle verticale (1..n)
A = couleur (d0..d3) et orientation (d4..d6) du texte
L'orientation du texte (d4..d6) est codée ainsi :

- 0 : Est
- 1 : Sud-Est
- 2 : Sud
- 3 : Sud-Ouest
- 4 : Ouest
- 5 : Nord-Ouest
- 6 : Nord
- 7 : Nord-Est

En sortie

A est modifié

Exemple

```
        LXI  H,TEXTE     ;HL pointe le texte à afficher
        MVI  D,32N       ;abscisse d'affichage
        MVI  E,100N      ;ordonnée d'affichage
        MVI  B,01N       ;échelle horizontale = 1
        MVI  C,01N       ;échelle verticale = 1
        MVI  A,61H       ;direction Nord, couleur 1
        CALL TEXTEG
        RET

TEXTE DB 'Texte à afficher', 0DH
```

Note

Le sous-programme `TEXTEG` ne vérifie pas les échelles ni le dépassement du texte de la zone graphique. L'affichage peut être très déformé voire illisible si les paramètres sont erronés.

LIREPIX

Fonction

Lecture de la couleur d'un pixel

En entrée

D = abscisse du pixel
E = ordonnée du pixel

En sortie

A contient la couleur du pixel
Il faut appliquer un masque à l'octet lu de façon à tenir compte du nombre de plans graphiques

Exemple

```
MVI  D,10N     ;abscisse du pixel
MVI  E,20N     ;ordonnée du pixel
CALL LIREPIX   ;obtenir la couleur du pixel
```

ECRIPIX

Fonction

Écriture d'un pixel

En entrée

D = abscisse du pixel
E = ordonnée du pixel
A = couleur de tracé (00H..10H)

En sortie

A est modifié

Exemple

```
MVI  D,10N     ;abscisse du pixel
MVI  E,20N     ;ordonnée du pixel
MVI  A,01H     ;couleur du pixel
CALL ECRIPIX   ;tracé du pixel
```

Note

La couleur spéciale 10H permet d'inverser la couleur du pixel aux coordonnées indiquées

Les paramètres et indicateurs de la bibliothèque graphique

Les paramètres et indicateurs de la bibliothèque graphique sont listés dans le tableau suivant :

Nom	Taille	Utilisation
PTSVECT	2 octets	Définition sur 16 bits des points d'un vecteur

PTSVECT

Fonction

Définition sur 16 bits des points d'un vecteur

Utilisation

Par défaut, PTSVECT vaut 0FFFFH (tous les points sont tracés)
Certaines valeurs particulières permettent de tracer les types de lignes suivants :

- 0CCCCH : trait pointillé
- 0E4E4H : trait d'axe
- 0EEEEH : pointillés longs
- 0AAAAH : un point sur deux

Le bit de poids fort définit le premier point tracé
À chaque tracé, la valeur de PTSVECT est décalée de façon circulaire vers la gauche
PTSVECT n'est pas réinitialisé après chaque tracé de vecteur, ceci permet de générer des motifs géométriques
Les sous-programmes VECTVE, VECTHO et VECTVE utilisent PTSVECT

Bibliothèque de calcul binaire

Cette bibliothèque cible les machines A10, A6 et A6E équipées d'un coprocesseur numérique Am9512.

En ce qui concerne les fonctions transcendantes, logarithmes et exponentielles, cette bibliothèque est basée sur le livre *Approximations for Digital Computers* de Cecil Hastings.

C'est la bibliothèque la plus complexe pour Alcyane. Elle a été écrite initialement par Jean B. de MBC en 3 semaines. Je l'ai reprise sans la modifier. Certains noms de sous-programmes peuvent sembler étranges, mais ils correspondent aux noms utilisés par Alcybaz. Je n'ai pas souhaité les modifier.

La bibliothèque contient les fichiers suivants :

- ADB1.asm
- ADB2.asm
- ASCBIN.asm
- BCDBN.asm
- BINBCD.asm
- BNASCI.asm
- BNBCD.asm
- CLNBR.asm
- COMPAR.asm
- CONVBC.asm
- CONVBN.asm
- COPYDH.asm
- COREXP.asm
- CPLHL.asm
- CVBCBN.asm
- CVBNBC.asm
- CVBNIT.asm
- CVITBN.asm
- DDE10.asm
- DDEN.asm
- DHL10.asm
- HL10.asm
- HORNER.asm
- IN9512.asm
- INDBCD.asm
- INDI.asm
- MATHS.asm
- OU9512.asm
- PARAM.asm
- PI.asm
- PLUS1.asm
- RABS.asm
- RADD.asm
- RADIAN.asm
- RASIN.asm
- RATAN.asm
- RCOMPA.asm
- RDTR.asm
- REXPE.asm
- RINT.asm
- RLOGE.asm
- RPOWER.asm
- RSIN.asm
- RSQRT.asm
- RZHL.asm
- STATUS.asm
- STB.asm
- TABLAT.asm
- TABLEI.asm
- TABLES.asm
- TABLEX.asm
- TABLLG.asm
- VARIAB.asm

Les pages qui suivent décrivent uniquement les points d'entrée intéressants, pas les sous-programmes utilitaires. Pour cela, la plupart des fichiers contiennent des commentaires assez descriptifs.

Les descriptions des sous-programmes sont groupées par fonctions.

Le fichier principal de la bibliothèque

Le fichier `MATHS.asm` contient les directives `USE` qui constituent la bibliothèque mathématique. Il contient également les points d'entrée à définir par l'utilisateur qui capturent les erreurs générées par les calculs :

- `ERR12` : valeur supérieure à 9999 dans `INDBCD` ou valeur négative dans `INDI`.
- `ERR20` : dépassement de capacité.
- `ERR22` : division par zéro.

Les fonctions d'entrées / sorties et l'indicateur de format

Les fonctions `ADB1` et `ADB2` (ce sont les noms historiques utilisés par Alcybaz) ainsi que l'indicateur `PTFIX` permettent d'entrer et de sortir des valeurs numériques dans la bibliothèque mathématique sous forme lisible.

Des exemples détaillés d'utilisation de ces fonctions sont donnés dans le programme de test de la bibliothèque mathématique.

ADB1

Module

`ADB1.asm`

Fonction

Transforme un texte représentant une valeur numérique en nombre binaire en virgule flottante codé sur 8 octets

En entrée

DE pointe le début du texte représentant une valeur numérique
HL pointe sur la zone de 8 octets qui recevra le nombre binaire

En sortie

DE pointe le premier caractère qui suit le texte représentant la valeur numérique
HL pointe sur le début du nombre binaire
Tous les autres registres sont modifiés

Exemple

```
CONVER  LXI  D,NOMBRE  ;texte représentant le nombre
        LXI  H,RESUL   ;zone recevant le résultat
        CALL ADB1      ;conversion
        RET

NOMBRE  DB  '-1.765E-33', 0DH
RESUL   DS  8          ;contiendra B9 22 54 2C C1 56 D0 27
```

ADB2

Module

ADB2.asm

Fonction

Transforme un nombre en virgule flottante codé sur 8 octets en texte affichable

En entrée

BC pointe sur le nombre à convertir suivi d'un octet à 0FFH
HL pointe la zone de mémoire qui recevra le texte représentant le nombre
PTFIX définit le format de sortie (voir plus bas)

En sortie

HL pointe le début du texte correspondant au nombre converti
Tous les autres registres sont modifiés

Exemple

```
CONVER  LXI  B,PI      ;texte représentant le nombre PI
        LXI  H,RESUL   ;zone recevant le résultat
        MVI  A,0EH     ;14 chiffres de précision
        STA  PTFIX
        CALL ADB2      ;conversion
        RET

PI      DB  40H, 09H, 21H, 0FBH, 54H, 44H, 2DH, 18H
        DB  0FFH
RESUL   DS  32         ;contiendra le texte "3.1415926535898"
```

PTFIX

Module

VARIAB.asm

Fonction

Définition sur un octet du format de sortie des nombres par le sous-programme ADB2

Utilisation

Les bits de l'octet PTFIX sont définis de la façon suivante :

- d7 = 1 : sortie sous forme d'un nombre flottant avec exposant
- d7 = 0 : sortie (si possible) sous forme d'un nombre sans exposant
- d6 = 1 : troncature de la valeur numérique au nombre de décimales demandé
- d6 = 0 : arrondi de la valeur numérique au nombre de décimales demandé
- d5 et d4 : inutilisés
- d3 à d0 : nombre de décimales (0 à 14)

Exemple

Exemples de sorties de ADB2 selon PTFIX avec la valeur 1.296 :

- PTFIX = 0EH : 1.2960000000000
- PTFIX = 02H : 1.30
- PTFIX = 42H : 1.29
- PTFIX = 86H : 1.296000 E0

Note

PTFIX correspond aux fonctions FIXED et FLOAT d'Alcybaz

Les quatre opérations

Les quatre opérations arithmétiques peuvent être exécutées par ces fonctions :

Nom	Utilisation
PLUS1	Addition
MOINS1	Soustraction
MULT	Multiplication
SLATCH	Division

PLUS1

Module

PLUS1.asm

Fonction

Addition de deux nombres en virgule flottante pointés par HL et DE

En entrée

DE pointe l'opérande 1
HL pointe l'opérande 2

En sortie

HL pointe le résultat opérande 1 + opérande 2
Les registres sont préservés sauf A

Exemple

```
LXI  D,VALEUR1     ;pointeur sur l'opérande 1
LXI  H,VALEUR2     ;pointeur sur l'opérande 2
CALL PLUS1         ;addition [DE] + [HL]
```

MOINS1

Module

`PLUS1.asm`

Fonction

Soustraction de deux nombres en virgule flottante pointés par HL et DE

En entrée

DE pointe l'opérande 1
HL pointe l'opérande 2

En sortie

HL pointe le résultat opérande 1 – opérande 2
Les registres sont préservés sauf A

Exemple

```
LXI  D,VALEUR1     ;pointeur sur l'opérande 1
LXI  H,VALEUR2     ;pointeur sur l'opérande 2
CALL MOINS1        ;soustraction [DE] - [HL]
```

MULT

Module

`PLUS1.asm`

Fonction

Multiplication de deux nombres en virgule flottante pointés par HL et DE

En entrée

DE pointe l'opérande 1
HL pointe l'opérande 2

En sortie

HL pointe le résultat opérande 1 * opérande 2
Les registres sont préservés sauf A

Exemple

```
LXI  D,VALEUR1     ;pointeur sur l'opérande 1
LXI  H,VALEUR2     ;pointeur sur l'opérande 2
CALL MULT          ;multiplication [DE] * [HL]
```

SLATCH

Module

`PLUS1.asm`

Fonction

Division de deux nombres en virgule flottante pointés par HL et DE

En entrée

DE pointe l'opérande 1
HL pointe l'opérande 2

En sortie

HL pointe le résultat opérande 1 / opérande 2
Les registres sont préservés sauf A

Exemple

```
LXI  D,VALEUR1     ;pointeur sur l'opérande 1
LXI  H,VALEUR2     ;pointeur sur l'opérande 2
CALL SLATCH        ;division [DE] / [HL]
```

Les fonctions transcendantes

Les fonctions transcendantes sont listées dans le tableau ci-dessous :

Nom	Utilisation
ESIN	Calcul de sinus
ECOS	Calcul de cosinus
ETAN	Calcul de tangente
EASIN	Calcul d'arc sinus
EACOS	Calcul d'arc cosinus
EATAN	Calcul d'arc tangente
ERAD	Passage en mode radians
EDEG	Passage en mode degrés
EGRA	Passage en mode grades

ESIN

Module

PLUS1.asm

Fonction

Calcul du sinus d'un nombre en virgule flottante pointé par HL

En entrée

HL pointe l'opérande

En sortie

HL pointe le résultat
Tous les registres sont modifiés

Exemple

```
        CALL EDEG           ;mode degrés
        LXI  H,ANGL30       ;pointeur sur la valeur 30
        CALL ESIN           ;calcul de sinus 30°
        RET

ANGL30 DB   40H, 3EH, 00H, 00H, 00H, 00H, 00H, 00H
```

ECOS

Module

PLUS1.asm

Fonction

Calcul du cosinus d'un nombre en virgule flottante pointé par HL

En entrée

HL pointe l'opérande

En sortie

HL pointe le résultat
Tous les registres sont modifiés

Exemple

```
        CALL EDEG           ;mode degrés
        LXI  H,ANGL45       ;pointeur sur la valeur 45
        CALL ECOS           ;calcul de cosinus 45°
        RET

ANGL45 DB   40H, 46H, 80H, 00H, 00H, 00H, 00H, 00H
```

ETAN

Module

PLUS1.asm

Fonction

Calcul de la tangente d'un nombre en virgule flottante pointé par HL

En entrée

HL pointe l'opérande

En sortie

HL pointe le résultat
Tous les registres sont modifiés

Exemple

```
        CALL EDEG           ;mode degrés
        LXI  H,ANGL60       ;pointeur sur la valeur 60
        CALL ETAN           ;calcul de tangente 60°
        RET

ANGL60 DB   40H, 4EH, 00H, 00H, 00H, 00H, 00H, 00H
```

EASIN

Module

PLUS1.asm

Fonction

Calcul de l'arc sinus d'un nombre en virgule flottante pointé par HL

En entrée

HL pointe l'opérande

En sortie

HL pointe le résultat
Tous les registres sont modifiés

Exemple

```
        CALL EDEG           ;mode degrés
        LXI  H,R2S2         ;racine carrée (2) / 2
        CALL EASIN          ;calcul de l'arc sinus
        RET

R2S2    DB   3FH, 0E6H, 0A0H, 9EH, 66H, 7FH, 3BH, 0E3H
```

EACOS

Module

PLUS1.asm

Fonction

Calcul de l'arc cosinus d'un nombre en virgule flottante pointé par HL

En entrée

HL pointe l'opérande

En sortie

HL pointe le résultat
Tous les registres sont modifiés

Exemple

```
        CALL EDEG           ;mode degrés
        LXI  H,R3S2         ;racine carrée (3) / 2
        CALL EACOS          ;calcul de l'arc cosinus
        RET

R3S2    DB   3FH, 0EBH, 0B6H, 7AH, 0E8H, 58H, 4CH, 0B7H
```

EATAN

Module

`PLUS1.asm`

Fonction

Calcul de l'arc tangente d'un nombre en virgule flottante pointé par HL

En entrée

HL pointe l'opérande

En sortie

HL pointe le résultat
Tous les registres sont modifiés

Exemple

```
        CALL EDEG           ;mode degrés
        LXI  H,UN           ;1.0
        CALL EATAN          ;calcul de l'arc tangente
        RET

UN      DB   3FH, 0F0H, 00H, 00H, 00H, 00H, 00H, 00H
```

EDEG

Module

`RADIAN.asm`

Fonction

Passage en mode degrés pour les fonctions transcendantes

En entrée

Rien

En sortie

Le registre A vaut 01H
L'indicateur `MODTRG` est modifié

Exemple

```
CALL EDEG           ;passe en mode degrés
```

ERAD

Module

`RADIAN.asm`

Fonction

Passage en mode radians pour les fonctions transcendantes

En entrée

Rien

En sortie

Le registre A vaut 00H
L'indicateur `MODTRG` est modifié

Exemple

```
CALL ERAD          ;passe en mode radians
```

EGRA

Module

`RADIAN.asm`

Fonction

Passage en mode grades pour les fonctions transcendantes

En entrée

Rien

En sortie

Le registre A vaut 02H
L'indicateur `MODTRG` est modifié

Exemple

```
CALL EGRA          ;passe en mode grades
```

Note

Le mode grades est utilisé par les géomètres dont beaucoup ont été des clients de MBC

Les fonctions de conversion

Les fonctions de conversion sont listées dans le tableau ci-dessous :

Nom	Utilisation
`RTD`	Conversion de radians en degrés
`DTR`	Conversion de degrés en radians
`CONVFR`	Conversion d'un entier 16 bits en binaire flottant
`CONFRF`	Conversion d'un binaire flottant en entier 16 bits
`BINARY`	Conversion d'un nombre flottant BCD en nombre flottant binaire
`DECIMA`	Conversion d'un nombre binaire flottant en nombre BCD flottant

Il existe d'autres fonctions de conversion dans la bibliothèque. Elles sont rarement utilisées. Les commentaires présents dans les fichiers permettent de comprendre leur utilisation.

RTD

Module

PLUS1.asm

Fonction

Conversion de radians en degrés

En entrée

HL pointe l'opérande

En sortie

HL pointe le résultat
Tous les registres sont modifiés

Exemple

```
CALL RTD          ;convertit les radians en degrés
```

DTR

Module

PLUS1.asm

Fonction

Conversion de degrés en radians

En entrée

HL pointe l'opérande

En sortie

HL pointe le résultat
Tous les registres sont modifiés

Exemple

```
CALL DTR          ;convertit les degrés en radians
```

CONVFR

Module

PLUS1.asm

Fonction

Conversion d'un entier 16 bits non signé en nombre binaire flottant

En entrée

BC contient la valeur binaire non signée

En sortie

HL pointe le résultat
Tous les registres sont modifiés

Exemple

```
LXI  B,1234H
CALL CONVFR          ;convertit l'entier en flottant
```

CONVRF

Module

PLUS1.asm

Fonction

Conversion d'un nombre binaire flottant en entier 16 bits non signé

En entrée

HL pointe l'opérande

En sortie

BC contient le résultat
Tous les registres sont modifiés

Exemple

```
          LXI  B,VAL1234
          CALL CONVRF          ;conversion en entier dans BC
          RET

VAL1234 DB 40H, 93H, 48H, 00H, 00H, 00H, 00H, 00H
```

BINARY

Module

PLUS1.asm

Fonction

Conversion d'un nombre flottant BCD au format Alcyane en nombre binaire flottant

En entrée

HL pointe l'opérande

En sortie

HL pointe le résultat
Tous les registres sont modifiés

Exemple

```
          LXI  H,BCD1234
          CALL BINARY          ;conversion BCD en binaire
          RET

;1234 au format Alcyane
BCD1234 DB 04H, 13H, 34H, 00H, 00H, 00H, 00H, 00H
```

Note

Cette fonction de conversion n'est utile que pour convertir des fichiers contenant des données numériques BCD au format Alcyane en données numériques binaires

DECIMA

Module

`PLUS1.asm`

Fonction

Conversion d'un nombre binaire flottant en nombre BCD flottant au format Alcyane

En entrée

HL pointe l'opérande

En sortie

HL pointe le résultat
Tous les registres sont modifiés

Exemple

```
          LXI  H,BIN1234
          CALL DECIMA          ;conversion binaire en BCD
          RET

;1234 en format binaire
BIN1234 DB 40H, 93H, 48H, 00H, 00H, 00H, 00H, 00H
```

Note

Cette fonction de conversion n'est utile que pour convertir des fichiers contenant des données numériques binaires en données numériques BCD au format Alcyane

Les fonctions diverses

Les fonctions diverses sont listées dans le tableau ci-dessous :

Nom	Utilisation
INT	Partie entière
ESQRT	Racine carrée
EEXPN	Exponentielle
ELOGN	Logarithme naturel (ou népérien)
ELOG10	Logarithme décimal
EPOWER	Élévation à une puissance
RABS	Valeur absolue
RSGN	Signe

INT

Module

PLUS1.asm

Fonction

Calcul de la partie entière d'un nombre en virgule flottante

En entrée

HL pointe l'opérande

En sortie

HL pointe sur le résultat
Tous les registres sont modifiés

Exemple

```
          LXI  H,V12345
          CALL INT          ;partie entière de 1.2345
          RET

;Valeur 1.2345
V12345  DB 3FH, 0F3H, 0C0H, 83H, 12H, 6EH, 97H, 8DH
```

ESQRT

Module

PLUS1.asm

Fonction

Calcul de la racine carrée d'un nombre en virgule flottante

En entrée

HL pointe l'opérande

En sortie

HL pointe sur le résultat
Tous les registres sont modifiés

Exemple

```
          LXI  H,DEUX
          CALL ESQRT         ;calcul de racine de deux
          RET

;Valeur 2
DEUX    DB 40H, 00H, 00H, 00H, 00H, 00H, 00H, 00H
```

EEXPN

Module

PLUS1.asm

Fonction

Calcul de l'exponentielle (e^x) d'un nombre en virgule flottante

En entrée

HL pointe l'opérande

En sortie

HL pointe sur le résultat
Tous les registres sont modifiés

Exemple

```
          LXI  H,TROIS
          CALL EEXPN          ;calcul de e3
          RET

;Valeur 3
TROIS     DB 40H, 08H, 00H, 00H, 00H, 00H, 00H, 00H
```

ELOGN

Module

PLUS1.asm

Fonction

Calcul du logarithme naturel (népérien) d'un nombre en virgule flottante

En entrée

HL pointe l'opérande

En sortie

HL pointe sur le résultat
Tous les registres sont modifiés

Exemple

```
          LXI  H,EBIN
          CALL ELOGN          ;calcul de ln (e)
          RET

;Valeur de e
EBIN      DB 40H, 05H, 0BFH, 0AH, 8BH, 14H, 57H, 03H
```

ELOG10

Module

PLUS1.asm

Fonction

Calcul du logarithme décimal d'un nombre en virgule flottante

En entrée

HL pointe l'opérande

En sortie

HL pointe sur le résultat
Tous les registres sont modifiés

Exemple

```
          LXI   H,DEUX
          CALL ELOG10          ;calcul de log (2)
          RET

;Valeur 2
DEUX      DB 40H, 00H, 00H, 00H, 00H, 00H, 00H, 00H
```

EPOWER

Module

PLUS1.asm

Fonction

Calcul de l'élévation à la puissance d'un nombre en virgule flottante

En entrée

HL pointe l'opérande
DE pointe la puissance

En sortie

HL pointe sur le résultat [HL] ** [DE]
Tous les registres sont modifiés

Exemple

```
          LXI   H,DEUX
          LXI   D,TROIS
          CALL EPOWER         ;calcul de 2³
          RET

;Valeur 2
DEUX      DB 40H, 00H, 00H, 00H, 00H, 00H, 00H, 00H
;Valeur 3
TROIS     DB 40H, 08H, 00H, 00H, 00H, 00H, 00H, 00H
```

RABS

Module

RABS.asm

Fonction

Calcul de la valeur absolue d'un nombre en virgule flottante

En entrée

HL pointe l'opérande

En sortie

HL pointe sur le résultat
Tous les registres sont modifiés

Exemple

```
          LXI  H,MOINS1
          CALL RABS          ;calcul de |-1|
          RET

;Valeur -1
MOINS1  DB 0BFH, 0F0H, 00H, 00H, 00H, 00H, 00H, 00H
```

RSGN

Module

RABS.asm

Fonction

Calcul du signe d'un nombre en virgule flottante

En entrée

HL pointe l'opérande

En sortie

HL pointe sur le résultat :

- -1.0 si l'opérande est négatif
- 0.0 si l'opérande est nul
- +1.0 si l'opérande est positif

Exemple

```
          LXI  H,MOINS1
          CALL RSGN          ;calcul du signe (-1)
          RET

;Valeur -1
MOINS1  DB 0BFH, 0F0H, 00H, 00H, 00H, 00H, 00H, 00H
```

Bibliothèque imprimante

Cette bibliothèque cible les machines A6 et A6E équipées de ports parallèles.

Elle est constituée d'un seul fichier : `imprim.asm`.

Les fonctions de la bibliothèque imprimante

Les fonctions de la bibliothèque imprimante sont listées dans le tableau suivant :

Nom	Utilisation
IMPINIT	Initialisation du signal STB
IMP1CAR	Impression d'un caractère sans time-out d'attente du signal DONE
IMPCARTO	Impression d'un caractère avec time-out d'attente du signal DONE
IMPIMM	Impression immédiate d'un caractère sans attente de signal DONE
IMPDOTO	Attente du signal DONE avec time-out, effacement du signal ACK
IMPBUF0D	Impression d'un buffer terminé par 0DH
IMPBUFLG	Impression d'un buffer sur longueur
IMPSTAT	Lecture de l'état de l'imprimante
IMPPRET	Détection de l'état « prêt » de l'imprimante

IMPINIT

Fonction

Initialisation du signal STB à l'état haut
Cette initialisation est indispensable faute de quoi, l'imprimante pourrait rater le premier caractère imprimé

En entrée

Rien

En sortie

A est modifié

Exemple

```
CALL IMPINIT        ;initialisation du signal STB
```

IMP1CAR

Fonction

Impression d'un caractère avec attente du signal DONE sans time-out

En entrée

A contient le caractère à imprimer

En sortie

Tous les registres sont préservés

Exemple

```
MVI  A,'A'
CALL IMP1CAR        ;impression d'un caractère A
```

IMPCARTO

Fonction

Impression d'un caractère avec time-out d'attente du signal DONE

En entrée

A contient le caractère à imprimer

En sortie

CY = 0 si le signal DONE a été reçu
CY = 1 si un time-out a été détecté
Tous les registres sont préservés

Exemple

```
MVI  A,'A'
CALL IMPCARTO       ;impression d'un caractère
JC   ERRIMP         ;saut si time-out d'attente du DONE
```

IMPIMM

Fonction

Impression d'un caractère sans attente du signal DONE

En entrée

A contient le caractère à imprimer

En sortie

Tous les registres sont préservés

Exemple

```
MVI  A,'Z'
CALL IMPIMM         ;impression immédiate d'un caractère
```

Note

Cette fonction, associée à la fonction `IMPDOTO`, permet de réaliser un pseudo multi-tâches en plaçant ces fonctions dans une boucle

IMPDOTO

Fonction

Attente du signal DONE avec time-out, effacement du signal ACK

En entrée

Rien

En sortie

CY = 0 si le signal DONE a été reçu, le signal ACK est alors effacé
CY = 1 si le signal DONE n'a pas été reçu

Exemple

```
CALL IMPDOTO        ;attente du signal DONE
JC   ERRDONE        ;saut si DONE non reçu
```

IMPBUF0D

Fonction

Impression d'un buffer terminé par 0DH

En entrée

HL pointe sur le début du buffer à imprimer

En sortie

CY=0 si aucune erreur n'a été détectée
CY=1 si une erreur a été détectée
A est modifié
HL est modifié
BC et DE sont préservés

Exemple

```
        LXI  H,BUFTXT
        CALL IMPBUF0D       ;impression du buffer
        JC   ERRIMPR        ;saut si erreur d'impression
        RET

BUFTXT DB 'Texte à imprimer', 0DH
```

IMPBUFLG

Fonction

Impression d'un buffer défini par sa longueur

En entrée

HL pointe sur le début du buffer à imprimer
BC contient le nombre de caractères à imprimer

En sortie

CY = 0 si aucune erreur n'a été détectée
CY = 1 si une erreur a été détectée
Les registres D et E sont préservés, tous les autres sont modifiés

Exemple

```
        LXI  H,BUFTXT
        LXI  B,SIZE (BUFTXT)
        CALL IMPBUFLG       ;impression du buffer
        JC   ERRIMPR        ;saut si erreur d'impression
        RET

BUFTXT DB 'Portion de texte à imprimer'
```

IMPSTAT

Fonction

Obtention de l'état de l'imprimante

En entrée

Rien

En sortie

d1 de A = 0 si l'imprimante est en défaut (signal HWA)
d2 de A = 1 s'il n'y a plus de papier dans l'imprimante (signal PE)
d3 de A = 1 si l'imprimante est sélectionnée (signal SEL)
Les autres bits de A sont à 0
Les autres registres sont préservés

Exemple

```
CALL IMPSTAT        ;lecture de l'état
CPI  0AH            ;teste SEL=1, PE=0, HWA=1
JZ   ETATOK
```

IMPPRET

Fonction

Indicateur d'imprimante prête

En entrée

Rien

En sortie

CY = 0 si l'imprimante est prête
CY = 1 si l'imprimante n'est pas prête
A est modifié, les autres registres sont préservés

Exemple

```
CALL IMPPRET        ;demande l'état de l'imprimante
JC   PASPRETE       ;saut si l'imprimante n'est pas prête
```

Bibliothèque V24

Cette bibliothèque cible les machines A6 et A6E équipées de ports série gérés par les composants 8251 et 8253.

La bibliothèque V24 ne contient que deux fichiers :

- `V24.asm`
- `V24.inc`

Le fichier v24.inc

Le fichier `v24.inc` contient les constantes utilisées pour paramétrer la liaison V24 dans une structure `PARV24`.

Le fichier v24.asm

Le fichier `v24.asm` contient toutes les fonctions nécessaires pour émettre et recevoir des données sur la liaison V24.

Les fonctions de la bibliothèque V24

Les fonctions de la bibliothèque V24 sont listées dans le tableau suivant :

Nom	Utilisation
`PROGV24`	Paramétrage de la liaison V24 : vitesse, parité, bits de données et de stop
`V24EMI`	Émission d'un caractère
`V24EMISA`	Émission d'un caractère sans attendre la fin de l'émission
`V24FINEM`	Indicateur de fin d'émission d'un caractère
`V24EMITO`	Émission d'un caractère avec gestion de temps dépassé
`V24REC`	Réception d'un caractère
`V24PRETR`	Indicateur de caractère prêt à recevoir
`V24RECTO`	Réception d'un caractère avec gestion de temps dépassé
`EMIBUFCR`	Émission d'un buffer terminé par CR (0DH)
`EMIBUFLG`	Émission d'un buffer défini par sa longueur
`RECBUFCR`	Réception d'un buffer terminé par CR (0DH)
`RECBUFLG`	Réception d'un buffer défini par sa longueur

PROGV24

Fonction

Initialisation de la liaison série V24

En entrée

HL pointe sur une structure `PARV24`

En sortie

CY = 0 si l'initialisation s'est bien déroulée
CY = 1 si un paramètre a une valeur invalide
A est modifié

Exemple

```
PARAM  PARV24                ;bloc de paramètres V24

       INCLUDE v24.inc       ;constantes pour V24

INIT   MVI  A,VIT_1200       ;1200 bauds
       STA  PARAM.VITESSE
       MVI  A,BITS_8         ;8 bits
       STA  PARAM.BITS
       MVI  A,PAR_PAIR       ;parité paire
       STA  PARAM.PARITE
       MVI  A,STOP_2         ;2 bits de stop
       STA  PARAM.STOP
       LXI  H,PARAM
       CALL PROGV24          ;programmation interface V24
       JC   ERREUR           ;saut si erreur
```

V24EMI

Fonction

Émission d'un caractère avec attente de fin d'émission

En entrée

A contient le caractère à émettre

En sortie

A est modifié

Exemple

```
MVI  A,"Z"
CALL V24EMI
```

V24EMISA

Fonction

Émission d'un caractère sans attente de fin d'émission

En entrée

A contient le caractère à émettre

En sortie

Rien

Exemple

```
MVI  A,"R"
CALL V24EMISA
```

Note

Cette fonction émet un caractère puis rend la main au programme principal, permettant un fonctionnement pseudo multi-tâches en utilisant `V24FINEM`

V24FINEM

Fonction

Indicateur de fin d'émission de caractère

En entrée

Rien

En sortie

CY = 0 si l'émission est achevée
CY = 1 si l'émission est toujours en cours

Exemple

```
CALL V24FINEM
JC   FINI
```

Note

Cette fonction, utilisée avec `V24EMISA` permet de rendre la main au programme appelant tant que l'émission n'est pas achevée, autorisant un fonctionnement pseudo multi-tâches

V24EMITO

Fonction

Émission de caractère avec gestion de temps dépassé

En entrée

A = caractère à émettre

En sortie

CY = 0 si le caractère a été émis
CY = 1 si le temps d'émission est trop long

Exemple

```
MVI   A,"M"
CALL  V24EMITO
JC    DEPASST
```

V24REC

Fonction

Attente de réception d'un caractère

En entrée

Rien

En sortie

A contient le caractère reçu

Exemple

```
CALL  V24REC
CPI   "B"
JZ    RECEPB
```

V24PRETR

Fonction

Indicateur de caractère prêt à recevoir

En entrée

Rien

En sortie

CY = 0 s'il y a un caractère prêt à recevoir
CY = 1 s'il n'y a pas de caractère prêt à recevoir

Exemple

```
CALL  V24PRETR
JC    PASRECU
```

V24RECTO

Fonction

Réception d'un caractère avec gestion de temps dépassé

En entrée

Rien

En sortie

CY = 0 si A contient le caractère reçu
CY = 1 si le temps de réception est dépassé

Exemple

```
CALL V24RECTO
JC   DEPASST
```

EMIBUFCR

Fonction

Émission d'un buffer terminé par CR (0DH)

En entrée

HL pointe le buffer à émettre

En sortie

CY = 0 si l'émission s'est achevée normalement
CY = 1 si le temps d'émission d'un caractère est dépassé
HL est modifié
A est modifié

Exemple

```
      LXI  H,BUF
      CALL EMIBUFCR
      JC   DEPASST

BUF   DB 'Texte émis en V24', 0DH
```

EMIBUFLG

Fonction

Émission d'un buffer défini par sa longueur

En entrée

HL pointe le buffer à émettre
BC contient le nombre de caractères à émettre

En sortie

CY = 0 si l'émission s'est achevée normalement
CY = 1 si le temps d'émission d'un caractère est dépassé
HL est modifié
BC est modifié
A est modifié

Exemple

```
        LXI   H,BUF
        LXI   B,SIZE (BUF)
        CALL  EMIBUFLG
        JC    DEPASST

BUF     DB 'Texte émis par longueur'
```

RECBUFCR

Fonction

Réception d'un buffer terminé par CR (0DH)

En entrée

HL pointe le buffer de réception

En sortie

CY = 0 si le buffer a été reçu
CY = 1 si le temps de réception d'un caractère est dépassé
HL est modifié
A est modifié

Exemple

```
        LXI   H,BUF
        CALL  RECBUFCR
        JC    DEPASST

BUF     DS    200
```

RECBUFLG

Fonction

Réception d'un buffer défini par sa longueur

En entrée

HL pointe le buffer de réception
BC contient le nombre de caractères à recevoir

En sortie

CY = 0 si le buffer a été reçu
CY = 1 si le temps de réception d'un caractère est dépassé
HL est modifié
BC est modifié
A est modifié

Exemple

```
        LXI   H,BUF
        CALL  RECBUFLG
        JC    DEPASST

BUF     DS    200
```

Le programme de test des bibliothèques

Le fichier `testalcy.zip` contient tous les fichiers sources et le fichier projet. Il est conçu pour être décompressé dans le répertoire `C:\Alcyane`.

Le mode d'emploi des bibliothèques est décrit en détail dans le programme de test. Le programme est constitué des fichiers suivants :

- `testalcy_08224_00000.asm` : programme appelant les sous-programmes
- `testalcy.asm` : cœur du programme de test
- `menucla.asm` : sous-programme de test des fonctions du clavier
- `menudsq.asm` : sous-programme de test des fonctions disque
- `menuecr.asm` : sous-programme de test des fonctions écran
- `menumat.asm` : sous-programme de test des fonctions mathématiques
- `menuimp.asm` : sous-programme de test des fonctions imprimante
- `utils.asm` : sous-programmes utilitaires divers

L'assemblage du programme de test et des diverses bibliothèques se fait en utilisant le fichier projet : `testalcy.prj`. Ce fichier projet est un simple fichier de texte aisément modifiable au moyen d'un éditeur standard comme Notepad, par exemple.

Note : temporairement, il n'existe pas de test pour les fonctions V24.

Le fichier testalcy_08224_00000.asm

Ce fichier contient les directives `USE` permettant d'appeler le programme principal (`testalcy.asm`) et les sous-programmes de test des diverses bibliothèques. Le nom étrange de ce fichier permet de fabriquer un fichier binaire directement utilisable par import dans l'émulateur.

Le fichier testalcy.asm

Ce fichier est le cœur du programme de test. Il affiche un menu à l'écran et dirige vers les tests des diverses bibliothèques.

Le fichier menucla.asm

Ce fichier contient un test unique permettant la saisie d'un buffer pré-initialisé.

Le fichier menudsq.asm

Ce fichier contient les tests des fonctions des disquettes sélectionnables par un menu :

- Lecture d'un secteur,
- Écriture d'un secteur,
- Formatage de disque,
- Affichage du catalogue,
- Lecture d'un fichier,
- Écriture d'un fichier,
- Suppression d'un fichier.

Tous les tests ont lieu sur le disque 1.

Le fichier menuecr.asm

Ce fichier enchaîne les différents tests de l'écran :

- Remplissage de l'écran en mode 80 colonnes,
- Remplissage de l'écran en mode 128 colonnes,
- Défilement de l'écran vers le haut en mode 80 colonnes,
- Défilement de l'écran vers le haut en mode 128 colonnes,
- Défilement de l'écran vers le bas en mode 80 colonnes,
- Défilement de l'écran vers le bas en mode 128 colonnes,
- Défilement vers le haut sur une portion d'écran,
- Défilement vers le bas sur une portion d'écran,
- Affichage d'un bloc de texte défini par sa longueur,
- Effacement d'un bloc de lignes en mode 80 colonnes,
- Effacement d'un bloc de lignes en mode 128 colonnes,
- Affichage avec inversion vidéo,

À la fin de chaque test, un curseur clignotant apparaît en bas à droite de l'écran. L'appui sur une touche quelconque passe au test suivant. À la fin, le menu principal est réaffiché.

Le fichier menugra.asm

Ce fichier propose tout d'abord la saisie du nombre de plans graphiques de la machine (de 1 à 4) puis propose les tests suivants :

- Initialisation du fond de page,
- Tracé de vecteurs,
- Tracé de cercles,
- Remplissage de surface,
- Tracé de texte,
- Tracé de barres.

Le fichier menumat.asm

Ce fichier propose les choix suivants :

- Quatre opérations arithmétiques,
- Fonctions transcendantes,
- Logarithmes et exponentielle,
- Affichage des arrondis.

Le fichier menuimp.asm

Ce fichier propose les choix suivants :

- Affichage en clair de l'état de l'imprimante
- Impression d'un texte de démonstration

Le fichier utils.asm

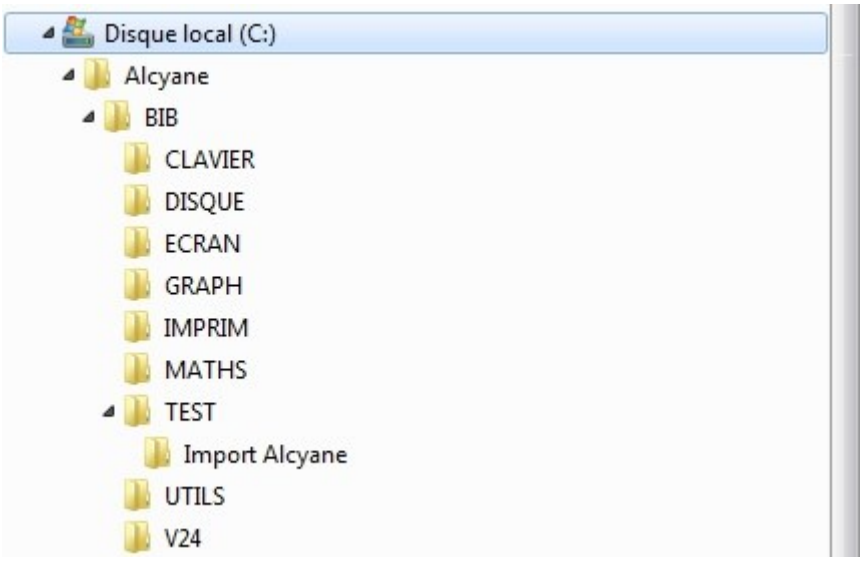
Ce fichier contient quelques sous-programmes utilisés par le programme principal.

Le fichier de projet testalcy.prj

Ce fichier contient :

- Le nom du fichier source principal et son chemin d'accès,
- Le nom du fichier binaire qui sera généré et son chemin d'accès,
- Les diverses options d'assemblage,
- Les chemins d'accès aux fichiers sources des bibliothèques et aux fichiers inclus.

Le fichier projet de démonstration suppose que les fichiers de test et les diverses bibliothèques sont placés dans l'arborescence suivante :



Compiler et exécuter le programme de test

Pour compiler le programme de test et l'exécuter avec l'émulateur :

1. Ouvrir le fichier `C:\Alcyane\BIB\TEST\testalcy.prj` dans l'assembleur.
2. Cliquer sur le bouton « Assembler ». Vérifier qu'aucune erreur n'a été détectée. La compilation dure un peu plus d'une seconde.
3. Copier le fichier `C:\Alcyane\BIB\TEST\testalcy_08224_00000.bin` dans le répertoire `C:\Alcyane\BIB\TEST\Import Alcyane`.
4. Lancer l'émulateur.
5. Insérer un disque vierge nommé « `DEMO` » dans le disque 1.
6. Formater ce disque nouvellement créé.
7. Importer les fichiers du répertoire `C:\Alcyane\BIB\TEST\Import Alcyane`. Les trois fichiers `DEMO01`, `FICTXT` et `TESTAL` doivent apparaître dans le catalogue.
8. Exécuter le test en tapant : `/1/TESTAL` au clavier.

Pour plus de détails, je vous invite à consulter le manuel de l'assembleur et celui de l'émulateur.