TÉLÉMATIQUE

Dragster

Manuel du langage Dragster

Septembre 1995

Table des matières

Introduction

Avertissement

Limites de garanties ou de responsabilité

Bien que JCA Télématique ait testé les programmes décrits dans ce manuel, ni JCA Télématique ni le concepteur des logiciels n'offrent de garanties, expresses ou tacites, concernant ce manuel ou les programmes qui y sont décrits, leur qualité, leur performance ou leur capacité à satisfaire à quelque application particulière que ce soit.

En conséquence, ces programmes et ce manuel sont vendus "tels quels", et l'acheteur supporte tous les risques en ce qui concerne leur qualité et leur fonctionnement. JCA Télématique et le concepteur des logiciels ne pourront en aucun cas être tenus pour responsables des préjudices directs ou indirects, de quelque nature que ce soit, résultant d'une imperfection dans les programmes ou le manuel, même s'ils ont été avisés de la possibilité que de tels préjudices se produisent. En particulier, ils ne pourront encourir aucune responsabilité du fait des programmes ou données mémorisées ou exploitées, y compris pour les coûts de récupération ou de reproduction de ces programmes ou données.

L'acheteur a toutefois droit à la garantie légale, dans les cas et dans la mesure seulement où la garantie légale est applicable nonobstant toute exclusion ou limitation. L'acheteur, pour bénéficier de cette garantie, doit renvoyer à JCA Télématique, dans le mois suivant l'achat, la carte de garantie du logiciel acheté dûment remplie.

Droits d'auteur

Ce manuel et les programmes qu'il décrit ont été déposés, tous droits réservés. Au terme de la législation des droits d'auteur, ce manuel et ces programmes ne peuvent être copiés, en tout ou partie, sans le consentement écrit de Philippe Boulanger, sauf dans le cadre d'une utilisation normale ou pour faire une copie de sauvegarde.

Ces exceptions n'autorisent cependant pas la confection de copies à l'intention d'un tiers, que ce soit ou non pour la vente, mais tout le matériel acheté (avec toutes ses copies de sauvegarde) peut être vendu, donné ou prêté à un tiers. Aux termes de la législation, copie signifie également traduction dans un autre langage ou format. Ces droits d'auteur couvrent de même tout ou partie des instructions Basic de Dragster* .

Edition de Septembre 1995

Contenu du manuel

Le présent manuel décrit le fonctionnement et l'utilisation du langage Basic de Dragster. Ce manuel de référence est destiné à être exploité par des personnes ayant des notions de programmation en Basic. Il n'essaie en aucun cas d'enseigner ni le langage Basic, ni une méthode de programmation ou d'analyse.

Le fonctionnement des interfaces Dragster (modems, Wit-Boost, Dragster Télétel) est décrit dans d'autres manuels.

Le fonctionnement et l'utilisation des programmes "DragsterEdit", "DragsterBoot" et "Dragster Startup" est décrit dans un autre manuel intitulé "Manuel d'utilisation de DragsterEdit".

Le fonctionnement et la création de routines externes Dragster sont décrit dans le manuel intitulé "Routines Externes Dragster".

Les informations présentes dans ce manuel sont susceptibles d'être modifiées dans un but d'amélioration. Pour vous tenir à jour ou nous faire part de vos remarques et suggestions, veuillez prendre contact avec:

JCA Télématique
Support Technique Dragster
4, rue Léon Bocquet
94100 Saint Maur des Fossés

Tél: 1-43 97 34 34
Fax: 1-43 97 17 17
Minitel: **3614** code **JCA**
AppleLink: **JCA.TELEMAT**
eMail: **jca@jca.fr**

A. Généralités sur le Langage Dragster

1. Le développement avec Dragster

1.1. Le langage

Le langage de Dragster est un Basic qui a été adapté aux besoins particulier des serveurs vidéotex.
Par rapport aux Basic “classiques”, le langage de Dragster diffèrent entre autre sur les points suivants:

- pas de numéros de ligne, ils sont remplacés par des étiquettes,
- seulement deux types de variables, chaine et entiers,
- tableaux à une seule dimension,
- fonctionnement multitâche,
- instructions spécifiques pour la gestion des serveurs vidéotex.

Les ajouts concernent des instructions empruntées à des langages plus évolués tels que le Pascal (WHILE/WEND).

Le concept “modulaire” des applications Dragster est aussi très différents des langages classiques, il s’apparente plus à des langages tels que HyperTalk (le langage d’HyperCard) où chaque carte comporte un script.

L’aspect multi-tâches est complètement intégré au langage et n’intervient que très peu dans le développement de votre application. Dans la plupart des cas, il suffit d’écrire son application sans se préoccuper de problèmes liés au multitâche.

Les problèmes liés à la communication sont gérés par Dragster, par exemple il suffit d’une seule instruction pour attendre un connexion (WAITCONNECT…).

1.2. Les différentes étapes de développement

Le développement d'une application sous Dragster comporte deux phases principales:

Phase 1: Construction et tests en mode interprété, ces derniers étant obligatoirement en version mono-voie. Cette construction et ces tests se font avec **DragsterEdit**.

Phase 2: Compilation (avec DragsterEdit) puis fonctionnement en version multi-voies à l’aide de DragsterBoot.

La mise au point de l’application se fait principalement en mode interprété, la mise au point étant beaucoup plus difficile en tâche de fond.

1.3. Concept de programmation de Dragster

Les applications "serveur Vidéotex" développées avec Dragster ont une structure modulaire obligatoire, ce qui n'est pas évident pour tous les Basics...

Le principe de base du langage Dragster est le suivant:

"A chaque écran Vidéotex correspond un code Basic qui se gèrera cet écran (affichages, saisies, contrôles, et enchaînements vers d'autres écrans)".

Une application correspond donc à un ensemble de modules qui s’enchainent les uns les autres.

Chaque module est composé d'un écran Vidéotex et d'un code Basic. Dans certains cas, seul l'écran Vidéotex ou seul le code Basic est utilisé.

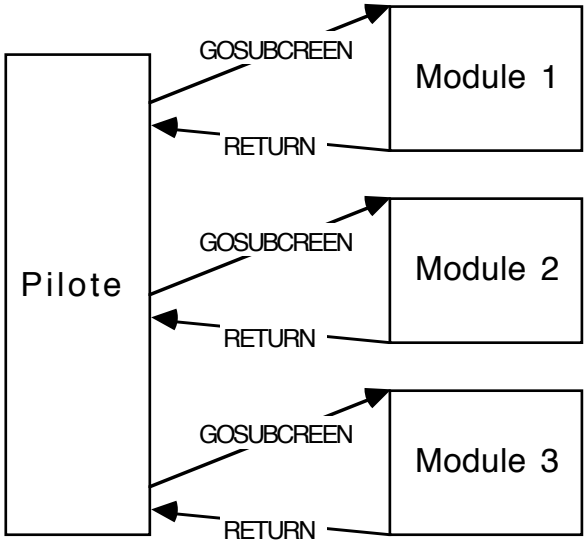
L'analyse de la structure du serveur passera donc par la définition de chacun des écrans visibles par l'utilisateur de l'application. Pour chacun de ces écrans seront déterminées les actions à utiliser au niveau du code Basic associé. Les variables Basic étant conservées d'un écran à l'autre, il est judicieux de faire un état strict des variables et de leur utilisation.

La décision de l'arborescence de l'application se fait à l'aide de deux ordres de base:

- **GOSUBSCREEN** "Nom_de_module",
  comparable au **GOSUB** étiquette à l'intérieur d'un module.
  On revient du module par un **RETURN**.

- **GOTOSCREEN** "Nom_de_module",
  comparable au **GOTO** étiquette à l'intérieur d'un module.
  On revient du module par un autre **GOTOSCREEN**.

Voici une structure possible pour votre application:



Chaque écran appelé peut, par exemple, faire appel à son module de Guide si nécessaire. Ce module de Guide n'a évidemment pas à être appelé au niveau du pilote, les guides étant fonction d'un écran ou d'une saisie spécifique.

2. Un langage unique pour différents modes de fonctionnement

En effet, le langage Dragster est toujours le même quel que soit le mode de fonctionnement de l'application écrite. Ces différences peuvent avoir leur importance même si un effort important a permis de standardiser au maximum le fonctionnement des application écrites avec Dragster.

2.1. Réseau téléphonique et réseau Transpac/Télétel

Les programmes créés avec le Basic Dragster fonctionneront indifféremment sur Réseau Téléphonique Commuté (RTC), Réseau Local, ou Transpac. Cependant, les modems intelligents utilisés par Dragster accomplissent des fonctions qui ne sont pas disponibles dans l'environnement Transpac. Le programmeur devra donc tenir compte de ces différences de fonctionnement s'il veut que son application ait un fonctionnement cohérent.

Les ordres du langage ayant un fonctionnement différent sous Transpac sont indiqués.

Par ailleurs, le fonctionnement même de Transpac diffère du fonctionnement sur le Réseau Téléphonique Commuté, les informations sont transmises par paquets et non octet par octet. Cette transmission n'est déclenchée que par certains caractères (par exemple, les touches de fonction du minitel telles que ENVOI ou SUITE). Les temps de transmission à travers le réseau ne sont pas non plus les même, il faut compter environ une seconde à travers Transpac alors que ces temps sont quasiment nuls sur RTC. Rassurez-vous, ces différences n'ont que très rarement une incidence sur votre application.

2.2. Mode interpreté et mode compilé

D'une manière générale, les cas d'erreurs n'arrêtent pas le déroulement du programme. Il est à la charge du programmeur de tester la fonction **ERROR** en retour des instructions critiques. Il existe cependant quelques cas où le déroulement d'un programme est stoppé en mode interpreté:

- indice d'un tableau trop élevé: l'interprèteur arrête le déroulement du programme.
- module non trouvé ou non exécutable lors d'un **GOTOSCREEN** ou **GOSUBSCREEN**
- module non trouvé lors d'un **DRAWSCREEN** ou **LOADSCREEN**

Le comportement dans ces cas d'erreurs est différent dans la version compilée:

- les dépassement d'indice de tableaux peuvent être détecté si vous le demandez lors de la compilation de votre application (voir le manuel d'utilisation de DragsterEdit). Si ceux-ci ne sont pas détectés, le Macintosh peut bomber ou faire planter le serveur.
- un module non trouvé lors des ordres **GOTOSCREEN**, **GOSUBSCREEN**, **LOADSCREEN** ou **DRAWSCREEN** font redémarrer immédiatement la tâche en erreur sur le module de démarrage. Pour remédier à ce problème, utiliser **DECSCREEN** pour déclarer cet écran manquant.

2.3. Mono-voie et multi-voies

Le fonctionnement d'une application en monovoie ou multi-voies diffère légèrement à plusieurs niveaux:

• les variables, elles peuvent être partagées par toutes les voies (ou tâches) ou exister en autant d'exemplaire qu'il y a de voies (ou tâches), (voir le chapitre sur les variables),
• les fichiers, des conflits peuvent apparaitre en passant de mono-voie à multi-voie (voir le chapitre sur les fichiers).

B. Contenu du langage Dragster

1. La syntaxe

Le Basic de Dragster possède une syntaxe proche de celle des Basic habituels.
Nous alons découvrir les principales caractéristiques de cette syntaxe.

1.1. Une ligne d'instructions

Les instructions du langage Dragster sont organisées en lignes. Chaque ligne peut contenir plusieurs instructions même si cela n'améliore pas forcément la lisibilité de votre programme.
Les différentes instructions présentes sur une ligne sont séparées par des ":", exemple:

```
CLS: DRAW: REM Affiche l'écran
```

Ce qui suit une instruction REM est toujours considéré comme une remarque.

1.2. Pas de numéros de lignes mais des étiquettes

Le Basic Dragster ne possède pas de numéros de lignes.
Pour permettre l'utilisation des instructions GOTO et GOSUB, ces numéros de lignes ont été remplacés par des "étiquettes", exemple:

```
DEBUT:
  CLS: DRAW: REM Affiche l'écran
  RESETZONES
  ZONE 22,20,10,X$,7

SAISIE:
  X$=""
  WAIT 1
  ON KEY GOTO TIME, DEBUT, ENVOI
  GOTO SAISIE

TIME:
  …
```

Dans cet exemple nous avons mis en caractères **gras** les étiquettes.
Pour définir une étiquette à un endroit du programme, celle-ci doit être seule sur une ligne et suivie de ":" (Ex: **DEBUT**:).
Pour appeler une étiquette, il suffit de mettre son nom (Ex: ON KEY GOTO **TIME**, **DEBUT**…).

1.3. Les IF/ENDIF, FOR/NEXT, WHILE/WEND

Contrairement à beaucoup de Basic, les instructions qui suivent un IF peuvent se trouver sur plusieurs lignes, la fin du IF est donnée par un ELSE ou un ENDIF.
Si vous oubliez un ENDIF, l'analyse de syntaxe vous indiquera une erreur.

Les structures IF/ELSE/ENDIF FOR/NEXT et WHILE/WEND peuvent être imbriquée, mais ne peuvent pas être croisées, exemple:

```
FOR A=1 TO 10
  B=B+A
  IF A<10: NEXT: ENDIF
```

Le NEXT se trouve dans un IF/ENDIF, ceci est incorrect !

Par ailleurs, les NEXT ne spécifient pas sur quelle variable ils agissent, ceci est automatique.
Les instructions BREAK et CONTINUE sont là pour éviter des problèmes de sortie de boucle.

2. Les variables

Dans ce chapitre, nous allons découvrir les différents types de variables de Dragster:

• variables numériques ou chaines
• variables partagées ou non
• tableaux

2.1. Variables numériques ou chaines

Le langage Dragster comporte deux types principaux de variable:

- les variables numériques entières signées sur 32 bits (occupation en mémoire: 4 octets).
exemple de nom de variable numérique: `A, X, MAVARIABLE`.
exemple de constante numérique: `12, 45670000, -432`

- les variables chaînes jusqu'à 255 caractères (occupation en mémoire: 256 octets).
exemple de nom de variable chaîne: `A$, X$, MAVARIABLE$`.
exemple de constante chaîne: `"toto", "MonFichier"`

Attention, un nom de variable ne peut pas dépasser 20 caractères. Ce nom commencera par une lettre (A..Z) et sera composé d'autres lettres ou chiffres (0..9).

Exemple: `A, B2$, C345, MAVAR3` sont des noms de variables.
`2A$` n'est pas un nom de variable.

Remarque: un constante chaîne ne peut dépasser 64 caractères.

2.2. Tableaux de variables

Les variables numériques et chaines sont aussi disponibles sous la forme de tableaux (appelées aussi variables indicées). Les tableaux ne peuvent avoir qu'une seule dimension. L'indice maximum des tableaux est donné à l'aide de l'ordre **`DIM`**.

Ainsi, pour une variable déclarée: **`DIM`** `A(10)`, les valeurs possibles de l'indice iront de 0 à 9.

Attention à ne pas dépasser les bornes des indices de tableaux, ceci pourrait perturber le fonctionnement de votre application et faire bomber le Macintosh en mode compilé.

2.3. Variables partagées ou non

Les variables sont par défaut locales à une tâche de l'application Dragster. Pour les communications entre voies, un concept de variables partagées à été développé. L'ordre **`SHARED`** permet ainsi de déclarer des variables partagées.

exemple: **`SHARED`** `MAVAR, COMVAR(1),A$`
permettra à toutes les tâches de travailler sur ces variables communes, qui restent des variables Basic classiques.

Afin de faciliter d'éventuelles communications et/ou synchronisations sur les variables partagées, des ordres spécifiques ont été créés: **`PEND`**, **`POST`** et **`REQUEST`**.

Ces variables partagées n'ont réellement d'utilité que dans une utilisation compilée multi-voie.

3. Les expressions

Les expressions dans Dragster sont des combinaisons d'opérateurs et d'opérandes. Les opérandes peuvent être des expressions, des variables, des constantes ou des fonctions.

Les opérateurs peuvent être divisé en trois classes:

- les opérateurs arithmétiques: **-, +, *, /, MOD**
- les opérateurs relationnel: **<, >, <=, >=, <>, =**
- les opérateurs logique: **NOT, OR, AND**

Les expressions sont toujours évaluées de gauche à droite, en prenant compte de la précédence des opérateurs, si cette précédence n'est pas exprimée clairement à l'aide de parenthèses. La précédence veut que certains opérateurs se trouvant sur un même expression soient évalués les premiers. L'ordre d'évaluation est alors le suivant:

- négation unaire (-), négation booléenne (**NOT**)
- multiplication, division, modulo (*, /, **MOD**) et le ET booléen (**AND**)
- addition, soustraction et OU booléen (**OR**)
- opérateurs relationnels (=, <, >, >=, <=, <>)

La syntaxe des expressions vous interdit cependant de créer des expressions illisibles: `(A = B) OR (A = C)` est une expression valide, mais `A = B OR A = C` est une expression invalide. Si l'analyseur de syntaxe vous indique une erreur, "parenthésez" l'expression, elle deviendra juste et lisible.

Exemple:

```
IF A=0 OR B=1: …
```
est refusé pas l'analyseur

```
IF (A=0) OR (B=1): …
```
est accepté

Les opérateurs disponibles sur les expressions chaîne sont les opérateurs relationnels (=, <, >, >=, <=, <>) et la concaténation (+).

Il est possible de faire intervenir des fonctions dans les expressions.

Exemple:

```
LEN(A$)*2+INSTR(A$,B$,1)

VAL(MID$(A$,INSTR(A$,"0",1),255)
```

4. La gestion des fichiers

Le Basic de Dragster gère par défaut des fichiers de type TEXT. Ces fichiers sont donc récupérables et modifiables par la quasi totalité des applications fonctionnant sur Macintosh.

La gestion de fichiers comporte les fonctions de base suivantes:

- création
- suppression
- ouverture
- fermeture
- écriture
- lecture
- positionnement dans le fichier (accès séquentiel)

Certaines intructions agissent sur le fichier lui même, d'autres sur les données contenues dans le fichier et d'autres sur la position courante dans le fichier.

Deux modes de positionnement différents existent dans la gestion des fichiers de Dragster:

- l'accès séquentiel
- l'accès direct

4.1. Instructions générales de gestion des fichiers

Création d'un fichier

La création d'un nouveau fichier de fait à l'aide de CREATE.

Exemple:

```
CREATE "MonFichier"
```

Si vous essayez de faire un **CREATE** d'un fichier qui existe déjà, rien ne se passe (en particulier, le fichier n'est pas remis à zéro).

Supression d'un fichier

La suppression d'un fichier de fait avec l'instruction KILL. KILL supprime le fichier et les données qu'il contient.
Si vous ne désirez que remettre un fichier à zéro, il est préférable d'utiliser l'instruction SETEOF pour forcer la taille du fichier à 0.

Ouverture d'un fichier

L'ouverture d'un fichier est faite par l'ordre **OPEN**. Cet ordre permet d'associer un **numéro logique** au fichier ouvert, de manière à ne spécifier le nom du fichier qu'à l'ouverture de celui-ci. Les autres instructions de la gestion de fichiers utiliseront ce numéro logique pour référencer un fichier ouvert. Les numéros logiques disponibles vont de 1 à 12, ce qui veut dire que pour une voie, à un instant donné, douze fichiers pourront être ouverts simultanément.

Exemple:

```
OPEN 1,"MonFichier"
```
ouvre le fichier "MonFichier" et lui associe le numéro logique 1.

Si on veut par la suite fermer ce fichier, l'ordre utilisé sera:

```
CLOSE 1
```

Remarques:
OPEN ouvre le fichier et se positionne au début de celui-ci, même si celui-ci contient déjà des informations.
Attention, **OPEN** ne crée pas le fichier si celui-ci n'existe pas. Vous devez créer les fichiers avec l'ordre **CREATE**.

Fermeture d'un fichier

Comme vous venez de le constater dans l'exemple précédent, l'instruction **CLOSE** sert à fermer un fichier.

Lecture depuis un fichier

Les ordres permettant de lire des données depuis un fichier sont :

- **READ**, qui lit des variables chaîne et numérique dans le fichier. Les valeurs sont séparées dans le fichier par des TAB (**CHR$**(09)) ou par des CR (**CHR$**(13))
- **BREAD**, qui lit de un à quatre octets dans le fichier et met la valeur correspondante dans une variable numérique.

Ecriture dans un fichier

Les ordres permettant d'écrire dans un fichier sont:

- **WRITE**, qui écrit des expressions chaîne et numérique dans le fichier. Les expressions sont séparées par des TAB et le WRITE génère un CR après la dernière valeur.
- **BWRITE**, qui écrit de un à quatre octets dans le fichier, sans générer ni TAB ni CR.
- **FPRINT**, qui a le même effet que PRINT, mais sur un fichier.

4.2. Accès "séquentiel" ou accès "direct" ?

Avant de lire ou écrire dans un fichier, il faut savoir où le faire. Le positionnement sert à déterminer à quel endroit du fichier Dragster va lire ou écrire des informations. Le positionnement séquentiel permet de se positionner à l'octet près dans un fichier (SEEK). Il est possible de demander à Dragster où se trouve la position courante (FPOS), quelle est la dernière position du fichier (GETEOF) et aussi de savoir si on a atteint cette dernière position (EOF).

Prenons l'exemple suivant:

```
CREATE "Test"
OPEN 1,"Test"
WRITE 1,"JCA"
WRITE 1,"43973434"
```

Sur le disque, le fichier contient les données suivantes:

Position:	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Donnée:	J	C	A	cr	4	3	9	7	3	4	3	4	cr	
														^

Nous retrouvons notre "JCA" et notre "43973434" séparés par un "cr" automatiquement ajouté par l'instruction WRITE.

Si nous voulons relire ces informations, nous devons nous “positionner” au début c’est à dire à la position 0.

```
SEEK 1,0
```

Le contenu du fichier n’est pas changé, seule notre position à changer:

Position:	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Donnée:	J	C	A	cr	4	3	9	7	3	4	3	4	cr	

^

Maintenant nous allons relire la première information:

```
READ 1,A$
```

La position courante à changé, Dragster est maintenant prêt à lire la deuxième information (ou à écrire dessus !).

Position:	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Donnée:	J	C	A	cr	4	3	9	7	3	4	3	4	cr	

^

L’accès séquentiel est pratique pour lire et écrire des fichiers “texte”. Ces fichiers ne comportent aucun espace vide (ou “trou”) entre les informations. Pour conserver des adresses dans un fichier séquentiel , il suffira d’écrire les informations les unes à la suite des autres. Vous vous appercevrez rapidement qu’il devient difficile d’accéder “directement” à une adresse donnée et qu’il faut donc lire le fichier à partir du début pour accéder à une adresse.

L'accès direct se différencie de l'accès séquentiel par un **pré-positionnement** sur un enregistrement donné du fichier avant une lecture ou une écriture. Ce pré-positionnement est possible grâce aux instructions:

- **RLEN**, qui permet de fixer la taille des enregistrements pour un fichier. Cette instruction est normalement utilisée une seule fois, après l'ouverture du fichier.
- **RSEEK**, qui permet de se positionner sur le début d'un enregistrement donné, donc pour chaque lecture ou écriture d'un enregistrement.

Exemple:

```
READADHERENT:
  REM routine de lecture d’une fiche adhérent
  OPEN 1,"Adhérents"
  RLEN 1,200: REM chaque enregistrement fait 200 octets
  RSEEK 1, NUMADHERENT: REM on se positionne
  READ 1,NOM$,PRENOM$: REM on lit les infos
  READ 1,ADR1$,ADR2$
  CLOSE 1
  RETURN: REM retour à l'appelant
```

Chaque enregistrement a donc une taille FIXE. Il faut prévoir une taille d’enregistrement qui soit assez grande pour recevoir dans tous les cas les informations d’une fiche, en effet, si votre enregistrement est trop petit, les informations d’une fiche déborderont sur la fiche suivante et vous perdrez le début de la fiche suivante.

4.3. Utilisation des fichiers en multi-voies

Le fait que Dragster fonctionne en multi-voie peut occasionner des conflits sur les fichiers. Si

deux voies désirent écrire en même temps sur un même fichier, voici ce qui risque de se passer:

- la voie 00 exécute:

```
OPEN 1,"MonFichier"
SEEK 1,GETEOF(1)
WRITE 1,"Toto","Tata"
CLOSE 1
```

- pendant que la voie 02, par exemple, exécute

```
OPEN 4,"MonFichier"
SEEK 4,GETEOF(4)
WRITE 4,"Test","Essai"
CLOSE 4
```

Il y aura une haute probabilité que l'on ait dans le fichier quelque chose dans le style: "Toto Test Tata Essai", ce qui n'est pas le résultat désiré. Encore pire, pour les deux voies, le **WRITE** fonctionnerait mal.

Les instructions **LOCK** et **UNLOCK** évitent ce genre de problème. **LOCK** permet de réserver un fichier à une voie jusqu'au **UNLOCK** ou au **CLOSE** de ce fichier par la même voie.

Le problème précédent aurait été résolu en écrivant:

- pour la voie 00:

```
OPEN 1,"MonFichier"
LOCK 1: REM on reserve le fichier
SEEK 1,GETEOF(1): REM on se positionne à la fin
WRITE 1,"Toto","Tata"
UNLOCK 1
CLOSE 1
```

- pour la voie 02:

```
OPEN 4,"MonFichier"
LOCK 4: REM on reserve le fichier
SEEK 4,GETEOF(4): REM on se positionne à la fin
WRITE 4,"Test","Essai"
UNLOCK 4
CLOSE 4
```

Les **UNLOCK** ont été mis ici pour la pureté de la programmation, car **CLOSE** fait automatiquement un **UNLOCK** .

4.4. Volumes + dossiers + fichiers = HFS

Le système de fichier du Macintosh gère des volumes, des dossiers et des fichiers.

Les volumes correspondent, par exemple: à une disquette, un disque-dur, un serveur de fichier ou une partition. Un volume porte un nom et il peut y avoir plusieurs volumes “en ligne” sur un Macintosh.

Les dossiers permettent d’organiser un volume suivant une hierarchie, en effet, un dossier peut lui même contenir d’autres dossiers ou des fichiers. Il est possible d’avoir des dossiers “vides”.

Les fichiers sont les éléments qui sont directement gérés par la gestion de fichiers de Dragster.

Ce système d'organisation de volumes est appelé HFS (Hierarchical File System). Dragster offre quelques instructions pour avoir des informations sur la structure des volumes.

Pour savoir quels sont les volumes en ligne: **GETVOL$**

Pour connaître le contenu d'un volume ou d'un dossier: **GETFILE$**

Pour lire ou modifier les informations concernant un fichier: **GETFINFO** et **SETFINFO**.

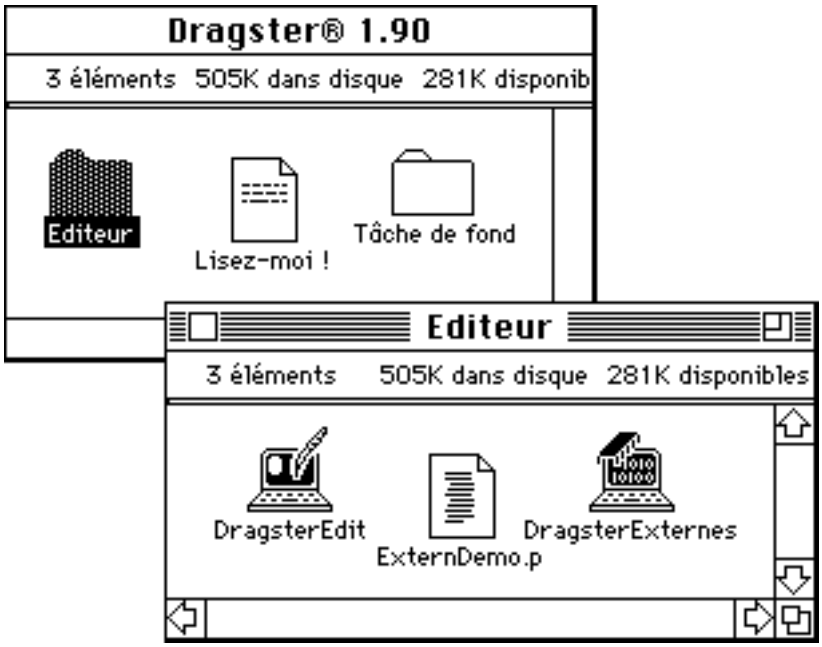
4.5. Noms de fichiers, chemins d'accès

Comme nous venons de le voir, un volume est organisé suivant une structure hiérarchique. Pour accéder à un fichier particulier depuis Dragster, il faut indiquer l'emplacement EXACT de celui-ci, c'est à dire: le nom du volume, les noms des différents dossiers et sous-dossiers intermédiaires, le nom du fichier. Cet emplacement exact est aussi appelé "chemin d'accès" complet.

Les différents noms de volume, dossiers et du fichier sont séparés entre eux par un deux-points ":". Ceci veut donc dire que ce caractère est interdit dans les noms de volumes, dossiers et fichiers.

Exemple de chemin d'accès:

Prenons par exemple la disquette "Dragster® 1.90" (celle-ci ne correspond pas à votre disquette Dragster), au niveau du Finder nous avons les fenêtres suivantes:



Le chemin d'accès du fichier "Lisez-moi !" est: "`Dragster® 1.90:Lisez moi !`"

celui du fichier "ExternDemo.p" est "`Dragster® 1.90:Editeur:ExternDemo.p`"

Remarques:

Les noms de volumes, de dossiers et de fichiers ne sont pas sensibles aux majuscules/minuscules, par contre les caractères accentués sont important.

```
MonFichier  =  MONFICHIER
```

mais

```
MesDonnées  ≠  MESDONNEES
```

Faites attention aux espaces au début et à la fin des noms de fichiers, en effet, certaines personnes ont la mauvaise habitude d'effacer un caractère en tapant un espace. Si vous supprimez le "s" de "Fichiers" en tapant un espace, celui-ce sera invisible au niveau du Finder, ceci est une cause très courante de problème d'ouverture de fichier.

5. La norme vidéotex, notions de présentation

Le labgage Basic de Dragster possède des instructions spécialement étudiées pour les besoins d'un serveur vidéotex. Ces instructions permettent d'utiliser les différentes ressources offertes par la norme Télétel.

Nous pouvons classer ces instructions en plusieurs groupes:

- gestion des attributs
- affichages
- saisies

5.1. La gestion des attributs

Le norme vidéotex Télétel permet d'afficher deux types de caractères: textes et mosaïques (graphiques). Ces caractères peuvent être modifiés par les attributs suivants:

Attribut	Texte	Graphique	Dragster
Inversion	Oui	Non	INVERSE
Souligné	sur un espace	Oui: (Disjoint)	UNDERLINE
Clignotement	Oui	Oui	FLASH
Coul. Caractère	Oui	Oui	FORECOLOR
Coul. Fond	sur un espace	Oui	BACKCOLOR

Pour le texte, certains attributs ne sont modifiables que sur des espaces.
Pour les graphiques seul l'inversion est indisponible.

Le passage du jeu texte au jeu graphique se fait avec l'instruction FONT.

5.2. Affichages

Les affichages peuvent se faire de différentes façons:

- utilisation de PRINT
- utilisation de MESSAGE
- envoi d'un écran vidéotex (DRAW, DRAWSCREEN, PRINTSCREEN)
- utilisation des zones de saisies

5.2.1. PRINT et MESSAGE

PRINT permet d'afficher n'importe quel informations sur le minitel. Cet affichage se fait avec les derniers attributs sélectionnés, et à la position courante.

Exemple:

```
LOCATE 1,1
FLASH 1
PRINT "Bonjour"
```

Ces instructions afficheront le mot "Bonjour" en ligne 1 colonne 1 en caractères clignotants.
Le locate remet les attributs en normaux.

MESSAGE permet d'afficher un message pendant un certain temps à une position donnée sur l'écran, puis d'effacer ce message. Cette instruction est très pratique, elle remplace avantageusement tout un ensemble de LOCATE, PRINT et autres instructions…

5.2.2. Affichage d'écrans vidéotex

Les écrans vidéotex permettent d'afficher rapidement et facilement des fonds pour des saisies ou peuvent même être l'unique information qui sera consultée sur le Minitel.

L'envoi d'un écran vidéotex peut se faire à l'aide de DRAW, DRAWSCREEN et PRINTSCREEN.
La différence entre ces instructions réside dans le fait que DRAW et DRAWSCREEN inclueront l'écran vidéotex en question dans le code compilé de votre application, alors que PRINTSCREEN chargera toujours l'écran vidéotex depuis un disque au moment de l'envoi.

PRINTSCREEN est plus intéressant que DRAWSCREEN pour les raisons suivantes:

- un changement d'écran ne nécessite pas de recompilation,
- le nombre d'écrans n'est pas limité, en effet, les écrans inclus lors de la compilation seront en permanence en mémoire même si il ne sont que très rarement utilisés,
- les écrans affichés par PRINTSCREEN peuvent être créé avec n'importe quel logiciel de composition vidéotex,
- le chargement d'un écran par PRINTSCREEN est très rapide, vous n'aurez pas de ralentissements dûs aux PRINTSCREEN utilisés dans votre application.

5.2.3. Gestion de zones de saisie

Les zones de saisies sont un des points forts de Dragster. Les instructions RESETZONES, ZONE et WAIT permettent de simplifier énormément la gestion d'un ensemble de zones de saisies.
L'instruction ZONE permet de définir les différentes zones de saisies (position, longueur, attributs), à chaque zone est attribué une variable qui contiendra ce qui a été saisi dans la zone correspondante.
L'instruction WAIT permet de déclencher la saisie, Dragster gère alors pour vous toutes les touches "classique" telles que CORRECTION, ANNULATION, SUITE, RETOUR.

Pour plus de détails, repportez vous à chacune des explications concernant ces instructions.

C. Les instructions du langage Dragster

A propos des instructions du langage Dragster

La liste des instructions qui composent le langage Dragster sont triées par ordre alphabétique. Vous trouverez une liste de ces instructions triées par catégorie (fichier, communication, etc) dans les Annexes.

Pour chaque instruction, sont indiqués les informations suivantes:

- le type d'instruction (fonction numérique, chaine, fichier, etc)
- son nom
- ce qu'elle fait
- sa syntaxe
- des exemples
- d'éventuelles remarques ou informations
- un renvoi éventuel vers d'autres instructions

Une étoile "*" derrière le type indique que l'instruction modifie la valeur courante du code d'erreur. Ce code d'erreur peut être obtenu en permanence par la fonction **ERROR**.

•*FN fonction numérique* **ABS**

Rend la valeur absolue de son paramètre

Syntaxe: **ABS**(expression_numérique)

ex: **ABS**(-4) rend 4
ABS(4) rend 4

•*FN fonction numérique* **AND**

Rend 1 si les deux opérandes sont non nuls, 0 sinon

Syntaxe: opérande1 **AND** opérande2

ex: 1 **AND** 2 rend 1
1 **AND** 0 rend 0

Ouvre le fichier spécifié, lui associe un numéro logique, et se positionne en fin de ce fichier.

Syntaxe: **APPEND** file_number,file_name

où file_number est une expression numérique
et file_name est une expression chaîne

ex: **APPEND** 1,"MonFichier"

NOTE: **ERROR** contient le code d'erreur de l'instruction.
ERROR=0 si tout est OK.

***ATTENTION**:*

A n'utiliser qu'en version monovoie ou en multivoies sur des fichiers qui ne risquent pas des conflits d'accès.

En version multivoies, utilisez la séquence suivante:

```
OPEN 1,"LeFichier"
LOCK 1
SEEK 1,GETEOF(1)
```

mais ne faites pas:

```
APPEND 1,"LeFichier"
LOCK 1
```

cette séquence est dangereuse car une autre voie a peut être écrit dans le fichier entre le **APPEND** et le **LOCK**, en quel cas vous ne seriez plus positionné à la fin du fichier.

•*FN fonction numérique* **ASC**

Rend la valeur ASCII du premier caractère de son paramètre

Syntaxe: **ASC**(expression_chaine)
(voir Table ASCII en Annexe)

ex: **ASC**("ABCD") rend 65

Voir aussi: **CHR$()**

•IV instruction **BACKCOLOR**

Génère les codes Vidéotex correspondant à un changement de la couleur des fonds de caractères

Syntaxe: **BACKCOLOR** color
où color est une expression numérique comprise entre 0 et 7
(Table de correspondance des couleurs en Annexe)

ex: **BACKCOLOR** 1: REM fond bleu
PRINT " Bonjour"
affiche "Bonjour" sur un fond bleu"

Voir aussi: **FORECOLOR**

Attention:

Pour valider la couleur de fond les textes doivent être porécédés d'un espace.

•IC instruction **BREAK**

permet de sortir de la boucle **WHILE** ou **FOR** courante

Syntaxe: **BREAK**

ex:

```
WHILE I > 0
  GOSUB MaRoutine
  IF ERROR
    BREAK: REM Sortie du While si erreur...
  ENDIF
WEND
```

Voir aussi: **CONTINUE**

•*IF* instruction* **BREAD**

Lit une variable numérique dans un fichier ouvert. Attention: la lecture est binaire et non ASCII. On spécifie le nombre d'octets à lire.

Syntaxe: **BREAD** file_number,variable,bytes_number

ex: **BREAD** 1,X,1: **REM** lit un octet du fichier 1 et le met dans X
BREAD 1,X,2: **REM** lit 2 octets du fichier 1 et les met dans X
BREAD 1,X,3: **REM** lit 3 octets du fichier 1 et les met dans X
BREAD 1,X,4: **REM** lit 4 octets du fichier 1 et les met dans X

Voir aussi: **BWRITE**

Remarque:

Cette instruction permet de lire un fichier octet par octet, ceci est pratique si vous devez lire des fichiers dont le format n'est pas du tout celui utilisé par Dragster.

•*IF* instruction* **BWRITE**

Ecrit une expression numérique dans un fichier ouvert.

Syntaxe: **BWRITE** file_number,expression_numérique,bytes_number

ex: **BWRITE** 1,X,1: **REM** écrit 1 octet de X dans le fichier 1
BWRITE 1,X,2: **REM** écrit 2 octets de X dans le fichier 1
BWRITE 1,X,3: **REM** écrit 3 octets de X dans le fichier 1
BWRITE 1,X,4: **REM** écrit 4 octets de X dans le fichier 1

Voir aussi: **BREAD**

Attention: l'écriture est binaire et non ASCII. On spécifie le nombre d'octets à écrire. Aucun séparateur n'est généré. Cette instruction permet de contrôler parfaitement ce qui est écrit dans un fichier.

•*IV instruction* **CANBLOCK**

Efface un rectangle sur l'écran, compris entre deux lignes et après une colonne fixée.

Syntaxe: **CANBLOCK** ligne1,ligne2,colonne

ex: **CANBLOCK** 10, 20, 1
efface de la ligne 10 à la ligne 20 à partir de la première colonne

Voir aussi: **CANEOL**

•*IV instruction* **CANEOL**

Efface la fin de la ligne derrière la colonne spécifiée

Syntaxe: **CANEOL** ligne,colonne

ex: **CANEOL** 10,12: **REM** efface derrière la colonne 12 de la ligne 10

Voir aussi: **CANBLOCK**

•*FC fonction chaîne* **CHR$**

Rend le caractère dont le code ASCII est égal à l'expression numérique en paramètre.

Syntaxe: **CHR$**(expression_numérique)
(voir Table ASCII en Annexe)

ex: **CHR$**(65) rend "A"

Voir aussi: **ASC()**

•*IF* instruction* **CLOSE**

Ferme le fichier dont le numéro logique est donné en paramètre

Syntaxe: **CLOSE** file_number

où file_number est une expression numérique correspondant au numéro logique d'un fichier ouvert.

ex: **CLOSE** 1: **REM** on ferme le fichier 1

NOTE: **ERROR** contient le code d'erreur de l'instruction.
ERROR=0 si tout est OK.

•*IV instruction* **CLS**

Efface l'écran du Minitel distant.

Syntaxe: **CLS**

ex: **CLS**: **DRAW**: **REM** on redessine l'écran

•*FN fonction numérique* **CONNECTED**

Répond 1 si le modem/CVC est connecté, 0 sinon.

Syntaxe: **CONNECTED**

ex:
```
IF CONNECTED: REM on redessine si connexion
  DRAW
ENDIF
```

•*IC instruction* **CONTINUE**

Permet de remonter au test de la boucle **WHILE** ou **FOR** courante. Dans le cas du **FOR**, l'indice est modifié avant le test.

Syntaxe: **CONTINUE**

ex:
```
FOR X = 1 TO 12 STEP 2
  GOSUB MaRoutine
  IF ERROR
    CONTINUE: REM X suivant, on saute le reste du for
  ENDIF
  GOSUB ...
  ...
NEXT
```

•*IF* instruction* **CREATE**

Permet de créer un fichier dont le nom est donné en paramètre le fichier sera de type Texte, donc lisible par tous les logiciels de traitement de texte disponibles

Syntaxe: **CREATE** file_name

Ex: **CREATE** "MonFichier"

Remarque: Si le fichier existe déjà, Dragster ne fera rien de particulier.

•*FC fonction chaîne* **CURPOS$**

Donne la position actuelle du curseur du minitel

Syntaxe: **CURPOS$**
La chaîne rendue a le format suivant:
1er caractère: CHR$(64+ligne)
2e caractère: CHR$(64+colonne)

Ex:
```
LI = ASC(CURPOS$) : REM LI = N° de ligne où se trouve le curseur
```

Remarque:

Cette fonction dépend du type de terminal connecté. Elle peut ne pas fonctionner correctement en cas de réponse lente (par exemple sur Transpac), ou si le terminal distant n'est pas un Minitel (micro-ordinateur en émulation). Il vaut mieux éviter d'utiliser cette fonction du BASIC Dragster.

•*IV instruction* **CURSOR**

Permet de rendre apparent ou invisible le curseur sur le Minitel distant.

Syntaxe: **CURSOR** expression_numérique
Si l'expression numérique est nulle, le curseur devient invisible, sinon il devient visible.

ex: **CURSOR** 1: **REM** on rend le curseur visible

•*FC fonction chaîne* **DATE$**

Rend une chaîne contenant la date courante.

Syntaxe: **DATE$**(0)

ex: **DATE$**(0) rend "14/06/86"

Voir aussi: **TIME$**, **SECS**,**DATE2SECS**,**SECS2DATE$**, **LOGTIME**

• *FN fonction chaîne* **DATE2SECS**

Convertit une chaîne du type "JJ/MM/AAhh:mm:ss", c'est à dire date et heure, en une valeur numérique.

Syntaxe: **DATE2SECS**(date_heure)

ex: NEWYEAR = **DATE2SECS**("01/01/8700:00:00")

voir aussi: **LOGTIME**, **SECS2DATE$**, **SECS**, **TIME$**, **DATE$(0)**

Remarque

Cette fonction associée à SECS2DATE$ permet de faire des calculs sur les dates (voir les exemples en annexe).

N'oubliez pas de donner les valeur "hh:mm:ss" même si elle ne servent pas dans votre calcul, en effet un DATE2SECS("01/01/90") rend une valeur incohérente.

•*IC instruction* **DECSCREEN**

Permet de déclarer l'utilisation d'un écran au compilateur.Cet ordre est INDISPENSABLE si l'application est destinée à être compilée et ci celle-ci utilise des appels dynamiques de modules. Un appel dynamique correspond à un GOTOSCREEN ou GOSUBSCREEN dont le paramètre n'est pas une constante (voir exemple).

Syntaxe: **DECSCREEN** constante_chaine

ex:
```
DECSCREEN "Guide1": REM declarations des guides
DECSCREEN "Guide2"
DECSCREEN "Guide3"
FOR X = 1 TO 3 STEP 1
      GOSUBSCREEN "Guide"+STR$(X): REM appel du guide choisi
NEXT
```

•*IC instruction* **DELAY**

Permet de suspendre l'exécution du programme pendant le nombre de 1/60e de seconde spécifié en paramètre.

Syntaxe: **DELAY** tempo
où tempo est une expression numérique

ex: **DELAY** 120: **REM** attente pendant 2 secondes

Attention Ne jamais faire de **DELAY** 0, ceci équivaut à un délai infini (ou presque).

Remarque:

DELAY peut servir à “donner la main” aux autres tâches dans une boucle ou un calcul assez long, pendant le DELAY les autres voies fonctionneront normalement et la voie mise en DELAY sera réactivée une fois le délai écoulé.

Voir aussi: **YIELDCPU**

•*FC fonction chaîne* **DELSPCL$**

Supprime les espaces (" ") du début de l'expression chaîne donnée en paramètre.

Syntaxe: **DELSPCL$**(expression_chaine)

ex: **DELSPCL$**(" ABCD") rend "ABCD"

Voir aussi: **DELSPCR$**

•*FC fonction chaîne* **DELSPCR$**

Supprime les espaces (" ") de la fin de l'expression chaîne donnée en paramètre.

Syntaxe: **DELSPCR$**(expression_chaine)

ex: **DELSPCR$**("ABCD ") rend "ABCD"

Voir aussi: **DELSPCL$**

• *IC* fonction chaîne* **DEQUEUE$**

Permet de récupérer le premier message en attente dans la queue de la tâche

Syntaxe: **DEQUEUE$**

ex:
```
REM on affiche tous les messages en attente
WHILE QUEUESIZE
  PRINT DEQUEUE$
WEND
```

Voir aussi: **ENQUEUE**, **QUEUESIZE**, **RESETQUEUE**

•*IV instruction* **DIAL**

Fait un appel en numérotation automatique sur le réseau téléphonique commuté ou sur Transpac.

Syntaxe: **DIAL** chaîne
Les caractères admis sont les chiffres de 0 à 9, ainsi que les caractères A et T qui ont la signification suivante:
A: attente de deux secondes environ.
T: attente de tonalité

ex: **DIAL** "T11": **REM** appel de l'annuaire électronique

Remarque:

Cette instruction ne fonctionne qu'avec les modems Dragster sur le réseau téléphonique.

•*IC instruction* **DIM**

Permet de déclarer la dimension d'un tableau numérique ou chaîne.

Syntaxe: **DIM** tableau(dimension),...
où tableau est le nom de la variable indicée, et dimension une constante numérique indiquant le nombre d'éléments de ce tableau.

ex: **DIM** A$(20),C(50),X(12)
DIM TOTO(11)

Remarque: Les indices pour le tableau TOTO iront de 0 à 10 soit 11 éléments.

•*IV instruction* **DISCONNECT**

Permet de déconnecter le Minitel distant, libérant ainsi la voie.

Syntaxe: **DISCONNECT**

ex:
```
IF NBTENTATIVE = 3: REM goodbye au pirate...
      DISCONNECT
      GOTO START
ENDIF
```

Voir aussi: **WAITCONNECT**, **CONNECTED**

•*IV instruction* **DRAW**

Permet d'envoyer l'écran Vidéotex associé au code Basic courant sur le Minitel distant.

Syntaxe: **DRAW**

ex: **CLS**: **DRAW**: **REM** on efface et on redessine...

Voir aussi: **DRAWSCREEN**, **PRINTSCREEN**

•IV instruction **DRAWSCREEN**

Permet d'envoyer l'écran Vidéotex spécifié en paramètre sur le Minitel distant

Syntaxe: **DRAWSCREEN** expression_chaine

ex: **DRAWSCREEN** "MonLogo"

NOTE: si le module n'existe pas, l'application redémarre automatiquement.
Remarque: Il est préférable d'utiliser **PRINTSCREEN** à **DRAWSCREEN**.

Voir aussi: **DRAW**, **PRINTSCREEN**

•IV instruction **ECHO**

Permet de passer en mode "Secret" ou non sur les saisies de caractères, par exemple pour saisir un code d'accès ou un mot de passe.

Syntaxe: **ECHO** expression_numérique

ex: **ECHO** 0: **REM** mode secret
ECHO 1: **REM** mode normal

Remarque: Cette instruction ne fonctionne que sur les modems Dragster.
Sur Télétel, il suffit d'utiliser des zones de saisie de caractères noirs.

• IC* instruction **END**

Arrête l'exécution d'un programme en mode interprété. Est équivalent à RESTART en mode compilé.

Syntaxe: **END**

ex:
```
IF A$="FIN"
  END
ENDIF
```

Voir aussi: **RESTART**

• *IC* Instruction* **ENQUEUE**

Permet d'envoyer à une autre tâche une chaîne de caractères. Cette chaîne arrivera dans la queue de messages de cette tâche. Les numéros de tâches valides vont de 1 à 65 (voies Vidéotex) et de 256 à 265 (tâches annexes). C'est typiquement une utilisation pour les messageries conviviales qui est ici visée. On peut aussi très bien imaginer de donner un ordre à une tâche annexe par le biais de cette queue. ERROR est positionné si la queue est trop pleine pour recevoir le message.

Syntaxe: **ENQUEUE** Numéro_de_tâche,Message

ex: **ENQUEUE** X,A$

Voir aussi: **DEQUEUE$**, **QUEUESIZE**, **RESETQUEUE**

•*IF* fonction numérique* **EOF**

Rend 1 si on est en fin du fichier spécifié, 0 sinon.

Syntaxe: **EOF**(file_number)
où file_number est le numéro logique d'un fichier ouvert

ex:
```
WHILE NOT EOF(1): REM tant qu'il y a quelque chose a lire...
  READ 1,A$,B$
  GOSUB MaRoutine
WEND
```

•*IC fonction numérique* **ERROR**

Rend le code d'erreur courant, qui a été positionné par une instruction précédemment exécutée. **ERROR** rend 0 si tout est OK, sinon le code correspond à une erreur dont vous pourrez trouver la signification en Annexe.

Syntaxe: **ERROR**

ex:
```
OPEN 1,"MonFichier"
IF ERROR: REM MonFichier n'existe pas, on le crée
      CREATE "MonFichier"
      OPEN 1,"MonFichier"
ENDIF
```

Voir aussi: **KEY**

•*IV instruction* **FLASH**

Permet de passer en mode "clignotant" sur le minitel distant.

Syntaxe: **FLASH** expression_numérique

ex: **FLASH** 0: REM mode normal
FLASH 1: REM mode clignotant

•*IV instruction* **FONT**

Permet de changer le jeu de caractère sur le minitel distant.

Syntaxe: **FONT** expression_numérique

ex: **FONT** 0: **REM** jeu normal
FONT 1: **REM** jeu semi-graphique

•*IC instruction* **FOR-NEXT**

Permet d'exécuter des boucles à l'aide d'une variable. A chaque **FOR** doit correspondre un **NEXT** unique qui délimite la fin de la boucle. Les **FOR** peuvent être imbriqués.

Syntaxe:
```
FOR Variable = exp_num1 TO exp_num2 [STEP exp_num3]
      instructions...
NEXT
```

ex:
```
FOR X = 2 TO 9 STEP 2
  PRINT X
NEXT
```

aura comme résultat:

2
4
6
8

Voir aussi: **BREAK**, **CONTINUE**, **WHILE-WEND**

Remarque: Après analyse de syntaxe, le **STEP** est automatiquement ajouté.

•*IV instruction* **FORECOLOR**

Génère les codes Vidéotex correspondant à un changement de la couleur des caractères.

Syntaxe: **FORECOLOR** color
où color est une expression numérique comprise entre 0 et 7
(Table de correspondance des couleurs en Annexe)

ex: **FORECOLOR** 7: **REM** caractère blanc

Voir aussi: **BACKCOLOR**

•*FC fonction chaîne* **FORMAT$**

Formate une chaîne selon 3 paramètres:
- la longueur
- la position d'une virgule éventuelle
- la justification: gauche, centrée, droite

Syntaxe: **FORMAT$**(long,virg,just,expression_chaine)

ex:
```
FORMAT$(6,2,1,"312")   rend "  3,12"   - just droite
FORMAT$(6,2,0,"312")   rend " 3,12 "   - centre
FORMAT$(6,2,-1,"312")  rend "3,12  "   - just gauche
FORMAT$(6,1,1,"312")   rend "  31,2"
FORMAT$(6,0,1,"312")   rend "   312"
FORMAT$(5,0,1,"312")   rend "  312"
FORMAT$(4,0,1,"abc")   rend " abc"
```

•*IF* fonction numérique* **FPOS**

Rend la position actuelle dans le fichier dont le numéro logique est donné en paramètre. Cette position est donnée en nombre de caractères depuis le début du fichier.

Syntaxe: **FPOS**(file_number)

ex: **FPOS**(1) rend la position dans le fichier 1

Voir aussi: **GETEOF**, **SEEK**

Même résultat que le **PRINT**, mais le résultat est envoyé dans le fichier dont le numéro logique est donné en paramètre.
Un caractère CR est généré à la fin du **FPRINT** si celui-ci ne se termine pas par un point-virgule.
Les virgules génèrent des espaces (même la première virgule), les points-virgules ne génèrent aucun caractère.

Syntaxe: **FPRINT** file_number ; expressions
ou
FPRINT file_number , expressions

ex: **FPRINT** 1,1+2,**FORMAT$**(5,0,1,"ABC")
écrit: `" 3  ABC"` suivi d'un CR

FPRINT 1;1+2;**FORMAT$**(5,0,1,"ABC");
écrit: `"3  ABC"` sans CR après.

Voir aussi: **WRITE**

• *IV instruction* **FRONTSCREEN**

Permet de commencer et d'arrêter le téléchargement de la page d'accueil qu'affiche le modem dans le cas ou le serveur n'est pas disponible.

Une fois le téléchargement commencé, tous les ordres qui normalement vont sur le minitel vont dans cette page d'accueil (DRAW, PRINT, etc...). Attention, la page d'accueil est limitée à 2048 caractères pour les Modems et interfaces Wit-Boost, et 800 octets pour les boîtiers DRAGSTER Télétel.

Syntaxe: **FRONTSCREEN** 0 fin de téléchargement
FRONTSCREEN 1 début de téléchargement

Important: n'oubliez pas d'arrêter le téléchargement !!!

ex:
```
FRONTSCREEN 1
DRAWSCREEN "Page/Accueil"
LOCATE 0,1: PRINT "Rappelez dans un instant"
FRONTSCREEN 0
```

Note: Cet ordre est utilisé généralement une seule fois au début de l'application, pour initialiser les modems.

Remarque

Sur les modems, si un écran a été téléchargé, l'appui sur la touche "CONNEXION/FIN" déconnectera directement le minitel distant sans passer par l'écran "Dragster". Ceci ne fonctionne qu'en position "Téléphone", pas en position Local.

• *IO* fonction chaîne* **GET$**

Rend le contenu du buffer d'entrée du modem (ou buffer d'entrée Transpac) ou le contenu du buffer d'entrée du deuxième port série du Macintosh

Syntaxe: **GET$**(Port)

Si port = 0, buffer modem (ou transpac)
Si port = 1, buffer 2e port série

ex: A$ = **GET$**(0)

Voir aussi: **GETPAQ$**, **SERCONFIG**, **OPENSER**

Rend la taille actuelle(en caractères) du fichier dont le numéro logique est donné en paramètre.

Syntaxe: **GETEOF**(file_number)

ex: **GETEOF**(1) rend la taille du fichier 1

Voir aussi: **GETFINFO**, **SEEK**, **APPEND**, **FPOS**

• *IF* fonction chaîne* **GETFILE$**

Permet de connaître les noms de fichiers à l'intérieur d'un dossier. Les fichiers sont numéroté de 1 à n. La fonction rend une chaîne vide si le fichier correspondant à un numéro n'existe pas et une chaîne éventuellement suivie de ":" (NomDossier:) si c'est un dossier.

Syntaxe: **GETFILE$**(Nom_du_dossier,Numero)

ex:

```
PRINT "Contenu du dossier Hd:FS"
REM 1er fichier
J = 1
A$ = GETFILE$("Hd:FS",J)
WHILE A$ <> ""
  IF RIGHT$(A$,1)=":"
    PRINT "Dossier ";LEFT$(A$,LEN(A$)-1))
  ELSE
    PRINT "Fichier ";A$
  ENDIF
  REM fichier ou dossier suivant
  J = J + 1
  A$ = GETFILE$("Hd:FS",J)
WEND
```

Attention Si le chemin d'accès donné est incorrect, c'est le contenu du plus haut niveau du disque dur qui sera rendu par **GETFILE$**.

Permet de récupérer des informations propres à un fichier ou un dossier.

Syntaxe:

GETFINFO File_Name,Type&Creator,DataLen,RsrcLen,CreateDate,LastModDate (fichier)

ou

GETFINFO Doss_Name,Type&Creator,nbElements,SauveDate,CreateDate,LastModDate (dossier)

où dans le cas d'un fichier:

File_name est le nom du fichier
Type&Creator est une variable chaîne qui contiendra en retour le type et le créateur du fichier,
DataLen et **RsrcLen** sont deux variables numériques contenant en retour la longueur du fichier (données et ressources),
CreateDate et **LastModDate** sont deux variables contenant en retour la date de création et de dernière modification du fichier.

ou bien dans le cas d'un dossier:

Doss_Name est le nom du dossier
Type&Creator est une variable chaîne qui contiendra en retour le type et le créateur du fichier,
NbElements est une variable numérique qui contient le nombre d'éléments présents dans le dossier
SauveDate, **CreateDate** et **LastModDate** sont trois variables contenant en retour la date de sauvegarde, de création et de dernière modification du fichier.

ex:

```
REM on affiche les noms des fichier de type TEXT du dossier Hd:Fs
PRINT "Fichier TEXT du dossier Hd:FS"
REM 1er fichier
J = 1
A$ = GETFILE$("Hd:FS",J)
WHILE A$ <> ""
  IF RIGHT$(A$,1)<>":": REM Ce n'est pas un dossier
    GETFINFO "Hd:FS:"+A$,TP$,A,B,C,D
    IF LEFT$(TP$,4)="TEXT"
      PRINT "Fichier ",J,":",A$
    ENDIF
  ENDIF
  REM fichier suivant
  J = J + 1
  A$ = GETFILE$("Hd:FS",J)
WEND
```

Voir aussi: **SETFINFO**, **GETFILE$**, **GETVOL$**

Permet d'identifier le minitel distant le paramètre permet de sélectionner une des trois identifications du minitel.

Syntaxe: **GETID$**(0) identification RAM1
GETID$(1) identification RAM2
GETID$(2) identification ROM

Voir aussi: **SETID**

Remarque Importante:

Cette fonction dépend du type de terminal connecté. Elle peut ne pas fonctionner correctement en cas de réponse lente (par exemple sur Transpac), ou si le terminal distant n'est pas un Minitel (micro-ordinateur en émulation), de plus les minitel actuels (1B) ne possèdent plus de RAMs !

Voici un équivalent de cette fonction pour lire la ROM:

```
FILTER 0
TRPRINT 1
PRINT CHR$(27);"9";CHR$(123);:REM Lecture ROM
X$ = "": N = 0
WHILE (INSTR(X$,CHR$(4)) = 0) AND (N < 5)
  DELAY 60
  X$ = X$ + GET$(0)
  N = N + 1
WEND
IF N >= 5: X$ = "": ENDIF
TRPRINT 0
FILTER 1

TRACE X$,ASC(X$),LEN(X$)
IF INSTR(X$,CHR$(1)): X$ = MID$(X$,INSTR(X$,CHR$(1)) + 1,255): ENDIF
```

• *IO* fonction chaîne* **GETPAQ$**

Attend un paquet de données sur le deuxième port série du Macintosh. Le paquet peut commencer par un caractère de synchronisation choisi et finir par un autre caractère de synchronisation choisi.

Syntaxe: **GETPAQ$**(TimeOut,MaxLen,StartSync,StopSync)
où TimeOut est le temps maximum d'attente d'un caractère,
MaxLen est la longueur maximale du paquet,
StartSync est la valeur Ascii du caractère de synchro de début de paquet (-1 si pas de synchro),
StopSync est la valeur Ascii du caractère de synchro de fin de paquet (-1 si pas de synchro).

ex:
```
REM on attend au plus 5 secondes un paquet d'au plus
REM 200 caractères, commençant par @ et finissant par CR
A$ = GETPAQ$(300,200,64,13)
```

Voir aussi: **GET$**, **SERCONFIG**, **OPENSER**

•*IF* fonction chaîne* **GETVOL$**

Permet de connaître les noms des volumes présent sur le bureau. Les volumes sont numéroté de 1 à n. La fonction rend une chaîne vide si le volume correspondant à un numéro n'existe pas.

Syntaxe: **GETVOL$**(index)

ex:
```
PRINT "Liste des volumes présents"
REM 1er volume
J = 1
A$ = GETVOL$(J)
WHILE A$ <> ""
  PRINT A$
  REM volume suivant
  J = J + 1
  A$ = GETVOL$(J)
WEND
```

•*IC instruction* **GOSUB**

Permet d'exécuter un sous programme dans le module courant. Le sous programme doit se terminer par un RETURN.

Syntaxe: **GOSUB** étiquette

ex: **GOSUB** MaRoutine
...

MaRoutine:
....
RETURN

•*IC instruction* **GOSUBSCREEN**

Permet d'exécuter un autre module que le module courant. On quitte le module appelé par un **RETURN**.

Syntaxe: **GOSUBSCREEN** module_name
où module_name est une expression chaîne

ex: **GOSUBSCREEN** "MonModule"

NOTE: si le module n'existe pas, l'application redémarre automatiquement.

NOTE: La fin du module appelé est considérée égale à un **RETURN**

Remarque:

Si le paramètre n'est pas une constante, vous devrez déclarez tous les noms d'écrans que vous utiliserez à l'aide de l'ordre **DECSCREEN**.

Voir aussi: **DECSCREEN, GOTOSCREEN**

•*IC instruction* **GOTO**

Permet de se “sauter” à une étiquette donnée dans le module courant.

Syntaxe: **GOTO** étiquette

ex: **GOTO** MaRoutine
...

MaRoutine:
....

•*IC instruction* **GOTOSCREEN**

Permet de se changer de module courant, l’éxécution se poursuivra au début du module donné en paramètre.

Syntaxe: **GOTOSCREEN** module_name
où module_name est une expression chaîne

ex: **GOTOSCREEN** "MonModule"

NOTE: si le module n'existe pas, l'application redémarre automatiquement.

Remarque:

Si le paramètre n’est pas une constante, vous devrez déclarez tous les noms d’écrans que vous utiliserez à l’aide de l’ordre **DECSCREEN**.

Voir aussi: **DECSCREEN, GOSUBSCREEN**

•*IC instruction* **IF-THEN-ELSE-ENDIF**

Permet une exécution conditionnelle. La partie **ELSE** est optionnelle. Le **ENDIF** est obligatoire. **ELSE** et **ENDIF** sont considéré comme des instructions. Les **IF**/**ENDIF** peuvent être imbriqués.

Syntaxe:

```
IF condition
      REM si condition <> 0
      instructions...
ELSE
      REM si condition = 0
      instructions...
ENDIF
ou
IF condition: ... :ELSE : ... : ENDIF
```

ex:

```
IF ERROR
      GOTO MessageErreur
ELSE
      PRINT "Ok"
ENDIF
```

•*IO* instruction* **INPUT**

Permet de saisir une variable

Syntaxe: **INPUT** message,variable
ou
INPUT variable

où message est une expression chaîne

ex: **INPUT** "Votre nom: ",NOM$

Important: **ERROR** contient la raison de sortie de l'ordre **INPUT**:
sortie par TimeOut/Deconnexion ou touche de fonction.

Voir aussi: **ZONE**

•*FN fonction numérique* **INSTR**

Rend la position d'une sous-chaîne dans une chaîne. rend 0 si la sous-chaîne n'est pas trouvée.

Syntaxe: **INSTR**(sous_chaine,chaîne)
où chaîne et sous_chaine sont des expressions chaîne.

ex: **INSTR**("abcdiafg","ia") rend 5
INSTR("abcdiafg","zz") rend 0

•*IV instruction* **INVERSE**

Permet de passer en mode "inverse" sur le minitel distant.

Syntaxe: **INVERSE** expression_numérique

ex: **INVERSE** 0: **REM** mode normal
INVERSE 1: **REM** mode inverse

• *FN fonction numerique* **KEY**

Utile pour les saisie, rend la valeur de **ERROR - 127.**

Syntaxe: **KEY**

ex:
```
WAIT 1
ON KEY GOTO XTIME,DESSIN,ENVOI
```

Supprime le fichier dont le nom est donné en paramètre

Syntaxe: **KILL** file_name
où file_name est une expression chaîne

ex: **KILL** "MonFichier"

Note: En version compilée, il est préférable de faire précéder les KILL d'un delay:

DELAY 1 : **KILL** "MonFichier"

•*FC fonction chaîne* **LEFT$**

Rend la partie gauche de la chaîne donnée en paramètre

Syntaxe: **LEFT$**(chaîne,longueur)
où chaîne est une expression numérique et longueur est une expression numérique indiquant le nombre de caractères à prendre à gauche de cette chaîne.

ex: **LEFT$**("abcdef",3) rend "abc"

Voir aussi: **MID$**, **RIGHT$**

•*FN fonction numérique* **LEN**

Rend la longueur de la chaîne donnée en paramètre

Syntaxe: **LEN**(chaîne)
où chaîne est une expression chaîne

ex: **LEN**(NOM$) rend la longueur du contenu de NOM$
LEN("toto") rend 4

•IV instruction LOADSCREEN

Permet de changer l'écran Vidéotex par défaut d'un code Basic (pour un **DRAW**, par exemple)

Syntaxe: **LOADSCREEN** module_name
où module_name est une expression chaîne

ex: **LOADSCREEN** "Masque1": **REM** masque de saisie 1

•IV instruction LOCATE

Permet de positionner le curseur sur le minitel distant.

Syntaxe: **LOCATE** ligne,colonne
où ligne et colonne sont des expressions numériques.
ligne peut être comprise entre 0 et 24
colonne peut être comprise entre 1 et 40

ex: **LOCATE** 0,1: **REM** ligne 0 colonne 1
LOCATE 24,1: **REM** dernière ligne

•IF* instruction LOCK

Spécifique à la version compilée multi-tâches. permet de réserver un fichier à la tâche qui le demande, ce qui évite les conflits si deux tâches écrivent dans ce fichier au même moment. Si le fichier était déjà **LOCK**é par une autre tâche, la tâche appelante est mise en attente jusqu'au déblocage du fichier par un **UNLOCK** ou un **CLOSE** de la tâche qui a réservé le fichier.

Tant qu'un fichier est "Locké" par une voie, il est impossible aux autres voies de lire, d'écrire dans ce fichier, ni de changer la taille du fichier (**SETEOF**).

Syntaxe: **LOCK** file_number
où file_number est une expression numérique donnant le numéro logique d'un fichier ouvert.

ex: **LOCK** 1

Voir aussi: **UNLOCK**

•*FN fonction numérique* **LOGTIME**

Rend la date et l'heure de connexion codée numériquement.

Syntaxe: **LOGTIME**

ex: **DISCONNECT**
REM Calcul du temps de connexion, en secondes
TEMPS = **ABS**(**SECS** - **LOGTIME**)

Voir aussi: **SECS**, **SECS2DATE$**, **DATE2SECS**, **TICKCOUNT**, **TIME$**, **DATE$**

ATTENTION !

LOGTIME ne fonctionne pas avec le boîtier Dragster Télétel, vous pouvez utiliser remplacer **LOGTIME** par un variable numérique que vous initialiserez lors de la connexion, exemple:

```
WAITCONNECT 0
MYLOGTIME = SECS
```

•*FC fonction chaîne* **LWC$**

Rend l'équivalent en "minuscules" de la chaîne donnée en paramètre.

Syntaxe: **LWC$**(chaîne)
où chaîne est une expression chaîne

ex: **LWC$**("BonJOur")
rend "bonjour"

Voir aussi: **UPC$**

•*IV instruction* **LOWER**

Met le Minitel en mode "enseignement" (minuscules par défaut) ou arrête ce mode.

Syntaxe: **LOWER** 0 mode normal (majuscules par défaut)
LOWER 1 mode enseignement (minuscules)

•*IV instruction* **MESSAGE**

Permet d'afficher un message à une position précise de l'écran, d'attendre un délai, et d'effacer le message affiché.

Syntaxe: **MESSAGE** ligne,colonne,délai,message
où ligne, colonne et délai sont des expressions numériques
et message est une expression chaîne.

ex:
```
MESSAGE 0,1,120,"Tapez ENVOI ou GUIDE"
```
fera s'afficher le message pendant deux secondes sur la ligne d'état (ligne 0).

•*FC fonction chaîne* **MESSDOWNLOAD$**

Rend le contenu du message de numéro donné en paramètre du modem affecté à la tâche. Sur le boîtier DRAGSTER Télétel, permet de récupérer les informations relatives à l'appel provenant de Télétel (voir Annexe concernant les particularités de programmation du Dragster Télétel). Ceci ne concerne pas les adaptateurs de type "NAMTEL".

Syntaxe: **MESSDOWNLOAD$**(message_number)
dans la version actuelle: message_number va de 0 à 10

•*IV instruction* **MESSUPLOAD**

Aucun effet dans la version Transpac envoie un message dans la messagerie du modem.

Syntaxe: **MESSUPLOAD** message_number,chaîne
dans la version actuelle: message_number va de 1 à 10

ex: **MESSUPLOAD** 4,"Rien à signaler"

•*FC fonction chaîne* **MID$**

Rend une sous-chaîne à l'intérieur d'une chaîne

Syntaxe: **MID$**(chaîne,position,longueur)
où chaîne est une expression chaîne et longueur et position sont des expressions numériques

ex: **MID$**("abcdefghi",2,4) rend "bcde"

Voir aussi: **LEFT$**, **RIGHT$**

•*FN fonction numérique* **MOD**

Rend le modulo de deux nombres, c'est à dire le reste de la division entière du premier opérande par le deuxième.

Syntaxe: opérande1 **MOD** opérande2

ex:
```
10 MOD 3      rend 1
 9 MOD 3      rend 0
 8 MOD 3      rend 2
```

•*FN fonction numérique* **MODNUMBER**

Rend le numéro du modem affecté à la tâche. Dans le cas de la version Transpac, ce numéro est le numéro logique de CVC.

Syntaxe: **MODNUMBER**

ex: A = **MODNUMBER** - 1

• *IF* instruction* **NEWFOLDER**

Permet de créer un dossier sur un des disques du Macintosh

Syntaxe: **NEWFOLDER** Folder_name

ex:
```
REM Création du dossier
NEWFOLDER "Hd:Dragster:Backup"
REM création d'un fichier dans le nouveau dossier
CREATE "Hd:Dragster:Backup:MonFichier"
```

•*IC instruction* **NEXT**

Voir: **FOR**

•*FN fonction numérique* **NOT**

Rend la valeur logique inverse de son argument: 0 si l'argument est non nul, 1 si l'argument est nul.

Syntaxe: **NOT** expression_numérique

ex:
```
IF NOT ERROR: REM equivalent a: IF ERROR = 0 ... ENDIF
```

Permet de sauter à une étiquette ou à un sous programme en fonction d'une expression numérique.

Syntaxe: **ON** expression **GOTO** etiq1,etiq2,...
ou
ON expression **GOSUB** etiq1,etiq2,...

La valeur de l'expression doit être comprise entre 1 et le nombre d'étiquettes, sinon l'instruction est ignorée.

ex:
```
SAISIE:
  WAIT 1
  ON KEY GOTO XTIME,ENVOI,GUIDE: REM sortie d'un WAIT
  GOTO SAISIE: REM sinon on refait un WAIT
```


Ouvre le fichier dont le nom est donné en paramètre et lui associe un numéro logique.

Syntaxe: **OPEN** file_number,file_name
où file_number est un numéro logique libre et file_name une expression chaîne

ex: **OPEN** 1,"MonFichier"

NOTE: le code d'erreur rendu par **ERROR** est positionné

NOTE: si le fichier n'existe pas, il ne sera pas créé, il faut donc vérifier **ERROR** dont voici les valeurs les plus fréquemment rencontrées:

-43 Fichier inexistant, créez le avec **CREATE**
-49 Fichier déjà ouvert (par Dragster ou une autre application)
-120 Chemin d'accès incorrect, un des noms de dossier intérmédiaire est incorrect.

Voir aussi: **APPEND**, **CLOSE, CREATE**

Ouvre et configure le deuxième port série du Macintosh et lui associe un numéro logique.

Syntaxe: **OPENSER** num_logique, vitesse, bits, parité, stop, protocole
où:

num_logique est un numéro logique libre (de 1 à 12)
pour les autres paramètres voir à **SERCONFIG.**

ex:
```
OPENSER 1,1200,7,2,1,2 : REM Config Minitel
IF NOT ERROR
        FPRINT 1;"Bonjour !"; : REM envoi sur port série de "Bonjour !"
        CLOSE 1 : REM fermeture du port série
ENDIF
```

NOTE: le code d'erreur rendu par **ERROR** est positionné

NOTE: si le fichier n'existe pas, il ne sera pas créé, il faut donc vérifier **ERROR** dont voici les valeurs les plus fréquemment rencontrées:

Voir aussi: **GETPAQ$**, **GET$**, **SERCONFIG**

•*FN fonction numérique* **OR**

Rend 1 si au moins un des deux opérandes est non nul. rend 0 si les deux sont nuls

Syntaxe: exp1 **OR** exp2
où exp1 et exp2 sont des expressions numériques

ex:
```
IF (A = 1) OR (C > 12)  ...   ENDIF
```

•*FN* fonction numérique* **PEND**

Spécifique à la version compilée multi-tâches. Permet de se mettre en attente, avec un délai maximum, sur une variable numérique. Si cette variable est non nulle, le résultat de **PEND** est le contenu de cette variable, et celle-ci est remise a zéro. On appelle une telle variable "boite aux lettres". Si la variable est nulle, une autre tâche devra la remplir avec une instruction **POST**. Si, après le délai maximum, la variable est toujours nulle, on sort du **PEND** avec une erreur de time out (rendue par la fonction **ERROR**). La variable utilisée dans **PEND** sera automatiquement "partagée" par toutes les tâches

Syntaxe: **PEND**(variable,delai_max)
où delai_max est une expression numérique

Attention, si délai_max est 0, il ne pourra jamais y avoir de Time Out.

ex:
```
A = PEND(MAILBOX1,600): REM 10 secondes de Time Out
```

Voir aussi: **POST**, **ERROR**, **SHARED**, **REQUEST**

•*IO* instruction* **POST**

Spécifique à la version compilée multi-tâches. Permet de mettre une valeur dans une boite aux lettres vide (nulle). si la boite aux lettres est pleine (non nulle), la tache sera mise en attente jusqu'à ce que cette boite aux lettres devienne vide (PEND d'une autre tâche), avec un délai maximum. La boite aux lettres est automatiquement une variable "partagée" par toutes les taches.

Syntaxe: **POST** variable,valeur,delai_max
où valeur et delai_max sont des expressions numériques

Attention, si délai_max est 0, il ne pourra jamais y avoir de time out.

ex: **POST** MAILBOX1,102,0
poste 102 dans MAILBOX1 sans Timeout

Voir aussi: **PEND**, **ERROR**, **SHARED**, **REQUEST**

Affiche ses paramètres sur le Minitel distant, à la position courante du curseur.
Une virgule entre deux expressions génère un espace, un point virgule ne génère pas d'espace.

Syntaxe: **PRINT** [expression,expression,...]

ex:
```
PRINT "Code d'erreur: ";ERROR
```
affichera : `Code d'erreur: 0` et passera à la ligne suivante

```
PRINT NB,"fiches trouvées";
```
affichera : `0 fiches trouvées` sans passage à la ligne

Voir aussi: **MESSAGE**

• IV* Instruction **PRINTSCREEN**

A le même effet que **DRAWSCREEN**, mais l'écran se trouve sur disque et non pas en mémoire. Le contenu de cet écran peut être modifié en cours d'exploitation du serveur sans avoir à recompiler l'application.
Le fichier n'a pas pas besoin d'être présent sur disque au moment de la compilation.
ERROR est positionné si le fichier n'existe pas.

Syntaxe: **PRINTSCREEN** Nom_de_fichier

ex:
```
WAITCONNECT 0
REM affichage de plusieurs écrans de présentation
REM qui s'appellent Hd:Sc:Present0, Hd:Sc:Present1,
REM Hd:Sc:Present2,...
J=0
WHILE NOT ERROR
      PRINTSCREEN "HD:Sc:Present"+STR$(J)
      DELAY 300:REM Temporisation entre écrans
      J=J+1:REM Ecran suivant...
WEND
```

• *IC* fonction numérique* **QUEUESIZE**

Donne le nombre de messages en attente dans la queue de la tâche.

Syntaxe: **QUEUESIZE**

ex:
```
REM on affiche tous les messages en attente
WHILE QUEUESIZE
  PRINT DEQUEUE$
WEND
```

Voir aussi: **ENQUEUE, DEQUEUE$, RESETQUEUE**

•*IF* instruction* **READ**

Permet de lire des variables dans un fichier ouvert

Voir aussi: **WRITE**, **FPRINT**

Syntaxe: **READ** file_number,var1,var2,...

ex: **READ** 1,A,NOM$,C$
lit les variables A, NOM$ et C$

NOTE: le séparateur de champ entre deux variables est soit TAB, soit CR.

•*IC instruction* **REM**

Permet de mettre des commentaires dans votre programme

Syntaxe: **REM** commentaire

NOTE: **REM** prend tous les caractères jusqu'à la fin de la ligne comme commentaire.

```
REM test: IF ERROR: GOTO XX: ENDIF
```
n'aura jamais aucun effet: Le IF fait partie du commentaire.

Permet de changer le nom d'un fichier

Syntaxe: **RENAME** old_name,new_name
où old_name et new_name sont des expressions chaîne
old_name est le nom du fichier à renommer
new_name est le nouveau nom pour ce fichier

ex: **RENAME** "MonFichier","SpoolFile"
change le fichier "MonFichier" en "SpoolFile"

Note: En version compilée, il est préférable de faire précéder les RENAME d'un delay:

DELAY 1 : **RENAME** "MonFichier","SpoolFile"

•*IC instruction* **REQUEST**

Spécifique à la version compilée multi-tâches. Un peu semblable à **PEND** pour les variables chaîne, **REQUEST** attend sur une variable chaîne avec un délai maximum. De plus, **REQUEST** attend que dans cette chaîne apparaisse les trois premiers caractères cités en argument.

Syntaxe: **REQUEST** delai_max,variable_chaine,chaine_attendue

ex: **REQUEST** 600,A$,"SYN" attendra que "SYN" apparaisse dans les trois premiers caractères de la variable A$, avec un délai maximum de 10 secondes.

• *IC* instruction* **RESETQUEUE**

Permet de remettre à zéro la queue de messages pour cette tâche. On fait généralement un **RESETQUEUE** après un **WAITCONNECT** ou juste après une entrée dans un service de messagerie conviviale.

Syntaxe: **RESETQUEUE**

ex:
```
WAITCONNECT 0
RESETQUEUE
```

Voir aussi: **ENQUEUE, QUEUESIZE, DEQUEUE$**

•*IV instruction* **RESETZONES**

Remet à zéro le nombre de zones de saisie du **WAIT**

Syntaxe: **RESETZONES**

ex:
```
RESETZONES: REM raz zones
REM déclaration des nouvelles zones
ZONE 10,12,8,A$,7: REM 1ere zone de saisie
ZONE 11,12,8,B$,7: REM 2e   zone de saisie
ZONE 12,12,8,C$,7: REM 3e   zone de saisie
WAIT 1: REM saisie des zones
```

Voir aussi: **ZONE**, **WAIT**

NOTE: RESETZONES ne vide pas le contenu des variables utilisées par ZONE ! C'est à vous de le faire.

•*IC instruction* **RESTART**

Redémarre la tâche en ayant préalablement fermé tous les fichiers. A utiliser si vous avez détecté une erreur grave, ou si il vous est trop fatiguant de revenir en haut de l'arborescence après une déconnexion.

Syntaxe: **RESTART**

NOTE: les variables de la tâche ne sont pas remises à zéro dans la version compilée.

•*IC instruction* **RETURN**

Revient d'un sous-programme appelé par **GOSUB** ou d'un module appelé par **GOSUBSCREEN**.

Syntaxe: **RETURN**

•*FC fonction chaîne* **RIGHT$**

Rend la partie finale de la chaîne donnée en paramètre

Syntaxe: **RIGHT$**(chaîne,longueur)
où chaîne est une expression numérique et longueur est une expression numérique indiquant le nombre de caractères à prendre à la fin de cette chaîne.

ex: **RIGHT$**("abcdef",3)
rend "def"

Voir aussi: **MID$**, **LEFT$**

•*IF* instruction* **RLEN**

Permet de spécifier la longueur d'enregistrement d'un fichier ouvert dont on donne le numéro logique. La valeur par défaut de la longueur d'enregistrement d'un fichier est 1.

Syntaxe: **RLEN** file_number,record_length
où file_number et record_length sont des expressions numériques

ex: **RLEN** 1,180: **REM** le fichier 1 a des enregistrements de 180 cars

Voir aussi: **RSEEK**, **FPOS**

•*IF* instruction* **RSEEK**

Permet de se positionner sur un enregistrement d'un fichier ouvert. Le premier numéro d'enregistrement dans un fichier est 0.

Syntaxe: **RSEEK** file_number,record_number
où file_number et record_number sont des expressions numériques

ex: **RSEEK** 1,20: **REM** enregistrement 20 du fichier 1

Voir aussi: **RLEN**, **FPOS**, **SEEK**

• *FN fonction numérique* **SECS**

Rend la date et l'heure codés sur une valeur numérique.

Syntaxe: **SECS**

ex:
```
REM memo des connexions
REM on mémorise la voie, le service, la date/heure de connexion,
REM et le temps de connexion
OPEN 1,"Hd:Sc:LogFile"
LOCK 1
SEEK 1,GETEOF(1)
WRITE 1,TASKNUMBER,SERVICE,LOGTIME,ABS(SECS-LOGTIME)
CLOSE 1
```

voir aussi: **LOGTIME, SECS2DATE$, DATE2SECS, annexes**

• *FN fonction chaîne* **SECS2DATE$**

Prend une valeur venant de **SECS, LOGTIME** ou **DATE2SECS** et la reconvertit en chaîne.

Syntaxe: **SECS2DATE$**(num_val)

ex:
```
REM on liste les connexions
REM on affiche la voie, le service, la date/heure de connexion,
REM et le temps de connexion
PRINT "Voie Service   Date   Temps"
PRINT "--------------------------"
OPEN 1,"Hd:Sc:LogFile"
WHILE NOT EOF(1)
      READ 1,TSK,SERVICE,LOGT,TEMPS
      PRINT TSK,"  ",SERVICE,"   ",SECS2DATE$(LOGT),TEMPS
      REM on cumule les temps
      TOT = TOT + TEMPS
WEND
CLOSE 1
PRINT "Temps total de connexion: ",TOT
```

voir aussi: **LOGTIME, DATE2SECS, SECS**

Permet de configurer le deuxième port série du Macintosh.
La lecture des données en entrée sur ce port doit se faire uniquement par **GET$** ou **GETPAQ$** (et surtout pas avec **READ** ou **BREAD**).
L'écriture sur ce port série se fait, après l'ouverture de celui-ci par **OPEN** 1,"**.**BOut", avec les ordres **FPRINT, WRITE**, ou **BWRITE.**

Syntaxe: **SERCONFIG** bauds,data,parity,stop,handshake

où bauds est la vitesse de transmission:
300, 600, 1200, 1800, 2400, 3600, 4800, 7200, 9600, 19200, 57600

data est le nombre de bits/caractère:
5, 6, 7, 8

parity indique le type de parité:
0 = pas de parité, 1 = parité impaire, 2 = parité paire

stop indique le nombre de bit de stop:
10 = 1 bit de stop, 15 = 1,5 bit de stop, 20 = 2 bits de stop

handshake indique le contrôle de flux:
0 = pas de contrôle, 1 = XON/XOFF, 2 = CTS-DTR

voir aussi: **GETPAQ$, GET$, OPENSER**

•*IV instruction* **SCROLL**

Met le Minitel en mode déroulement (mode des terminaux classiques) ou arrête ce mode déroulement.

Syntaxe: **SCROLL** 0 mode page
SCROLL 1 mode déroulement

•*IF* instruction* **SEEK**

Permet de se positionner sur un fichier ouvert. Le premier numéro de caractère dans un fichier est 0.

Syntaxe: **SEEK** file_number,position
ou file_number et position sont des expressions numériques

ex: **SEEK** 1,20: **REM** 20e caractère du fichier 1

Voir aussi: **RLEN**, **FPOS**, **RSEEK**, **GETEOF**

• *F* instruction* **SETEOF**

Permet de fixer la taille en caractères d'un fichier ouvert.

Syntaxe: **SETEOF** file_number,new_size
où file_number et new_size sont des expressions numériques

ex:
```
SETEOF 1,2000: REM taille du fichier 1: 2000 caracteres
SETEOF 1,0: REM raz du fichier 1
```

• *IF* instruction* **SETFINFO**

Permet de changer le type et le créateur d'un fichier

Syntaxe: **SETFINFO** File_Name,Type&Creator

ex:
```
REM création d'un fichier
CREATE "Hd:FS:toto"
REM on change son type à TEXT et son créateur à MYAP
SETFINFO "Hd:FS:toto","TEXTMYAP"
```

voir aussi: **GETFINFO, GETFILE$, GETVOL$**

•IV instruction SETID

Permet de mettre une identification dans la RAM du minitel distant.

Syntaxe: **SETID** 0,chaîne identification RAM1
SETID 1,chaîne identification RAM2

NOTE: une identification fait 15 caractères au maximum. Il est donc inutile d'envoyer plus de 15 caractères

Voir aussi: **GETID$**

• IC instruction SHARED

Spécifique a la version compilée multi-tâches. permet de déclarer au compilateur les variables "partagées" par toutes les tâches. Toute variable est présumée locale à une tâche sauf dans les cas suivant:

- elle est dans la liste d'une instruction **SHARED**
- elle est utilisée par **PEND**, **POST** ou **REQUEST**

Syntaxe: **SHARED** liste_de_variables

ex: **SHARED** MAILBOX1,SYNCHRO,CONVIV,TABLEAU(0)

Remarque: il suffit qu'un seul élément d'un tableau soit "partagé" pour que tout le tableau soit "partagé".

•IV instruction SIZE

Fixe la taille des caractères sur le Minitel distant pour le Print suivant.

Syntaxe: **SIZE** taille
où taille est une expression numérique comprise entre 0 et 3:
taille 0: taille normale
taille 1: double hauteur
taille 2: double largeur
taille 3: double hauteur

ex: **SIZE** 2: **REM** passage en double largeur

•*FC fonction chaîne* **SPC$**

Rend une chaîne contenant un nombre d'espaces donné en paramètre.

Syntaxe: **SPC$**(nombre_d'espaces)
où nombre_d'espaces est une expression numérique

ex: **SPC$**(5) rend la chaîne " " (5 espaces)

•*FN fonction numérique* **STARFLAG**

Indique si la saisie s'est arrêtée avec la zone courante se terminant par une étoile. Voir les recommandations aux partenaires Télétel (annexe 1).

Syntaxe: **STARFLAG**

ex:

```
SAISIE:
  WAIT 1
  ON KEY GOTO TIME,REP,ENVOI,GUIDE,SOMM,SUITE,RETOUR
  GOTO SAISIE

SOMM:
  IF STARFLAG
    GOTOSCREEN "SOMMGENE"
  ELSE
    GOTOSCREEN "SOMM"
  ENDIF
```

Voir aussi: **WAIT**, **INPUT**

•FC fonction chaîne **STATUS$**

Rend l'état courant de la voie. Permet de savoir quel type de matériel correspond à la voie courante.

Syntaxe: **STATUS$**
Cette fonction renvoie une chaîne qui a le format suivant:
1er car: M si modem, T si Transpac
2e car: C si connexion, F sinon
3e car: 0 si serveur, autre si service local (Modem)
4e car: L si minitel local, R si réseau téléphonique, T si Transpac
5e car: Phase du modem (Modem)
6e car: Dernière cause d'erreur du modem (Modem)
7e car: A si mode normal, R si modem inversé (Modem)

ex:
```
REM Lecture des infos Télétel si besoin...
IF MID$(STATUS$,4,1)="T"
  S$=MESSDOWNLOAD$(0)
ENDIF
```

•FC fonction chaîne **STR$**

Rend une chaîne correspondant à l'expression numérique donnée en paramètre.

Syntaxe: **STR$**(expression_numérique)

ex: **STR$**(123) rend la chaîne "123"

voir aussi: **VAL**

•FC fonction chaîne **STRING$**

Rend une chaîne contenant un caractère spécifié répété un certain nombre de fois.

Syntaxe: **STRING$**(nombre,caractère)
où nombre est une expression numérique et caractère une expression chaîne

ex: **STRING$**(5,"#") rend la chaîne "#####"
STRING$(5,"abcdef") rend la chaîne "aaaaa"

•*IV instruction* **SWMODEM**

Retourne le modem (et le minitel distant). C'est à dire que le serveur émettra à 75 bauds et le minitel distant à 1200 bauds.

Syntaxe: **SWMODEM**

•*IV instruction* **SYSPARM**

Met à jour certain des paramètres du modem:

- destruction des buffers du modem sur CLS ou non, 0 ou 1
- beep en cas d'erreur d'édition ou non, 0 ou 1
- délai, en secondes, au bout duquel le modem détecte une attente du serveur et affiche le message "Patientez" de 0 à 999
- délai, en secondes, au bout duquel le modem détecte une anomalie du serveur et affiche le message "Indisp." de 0 à 999

Syntaxe: **SYSPARM** cls_flag,bip_flag,delai1,delay2

ex: **SYSPARM** 1,0,10,60

•*FN fonction numérique* **TASKNUMBER**

Rend le numéro de tâche courant

Syntaxe: **TASKNUMBER**

ex:
```
IF TASKNUMBER=1
      GOSUB InitMailBoxes
ENDIF
```

Note: Cette fonction est utilisée, par exemple, pour savoir quelle tâche aura le droit d'initialiser des variables en début d'application.

Cette fonction est indispensable pour aiguiller les tâches annexes.

•*FN fonction numérique* **TICKCOUNT**

Rend le nombre de 1/60e de seconde écoulés depuis le démarrage du Macintosh.
(Utile pour chronométrer des opérations de façon précise).

Syntaxe: **TICKCOUNT**

Voir aussi: **DELAY**, **LOGTIME**, **SECS**

•*FC fonction chaîne* **TIME$**

Rend l'heure du jour sous la forme "hh:mm:ss".

Syntaxe: **TIME$**

ex: **TIME$** rend la chaîne "18:50:57" si il est 18h 50 mn et 57s

•*IV instruction* **TIMEOUT**

Fixe le temps maximum d'attente en **WAIT** ou **INPUT** pendant lequel un utilisateur n'a pas frappé de caractères. On estime alors que l'utilisateur s'est endormi, ou est allé boire son café...
La saisie est alors arrêtée et **ERROR** rendra alors la valeur 128 (TIMEOUT ou Déconnexion).

Syntaxe: **TIMEOUT** nombre_de_secondes
où nombre_de_secondes est une expression numérique

ex: **TIMEOUT** 120

NOTE: Il est déconseillé de programmer le **TIMEOUT** à une valeur inférieure à la minute. Les utilisateurs n'ont pas toujours un temps de réponse très rapide...

•*IO instruction* **TRACE**

--- effet visible seulement dans l'interprèteur

Même effet et même syntaxe que l'instruction **PRINT**, mais la sortie se fait sur la fenêtre "ERREURS/TRACES SYSTEME" de l'interprèteur Basic. Cette instruction est sans effet en version compilée. Très utile en débogage (et oui, c'est du français...)

•*IC instruction* **TRON**

--- effet visible seulement dans l'interprèteur

Mode espionnage de l'interprèteur: à chaque fois qu'une étiquette est rencontrée, elle est affichée dans la fenêtre ERREURS. A chaque changement de module, le nom du nouveau module est affiché dans la fenêtre ERREURS.

Syntaxe: **TRON**

Voir aussi: **TROFF**

NOTE: Le même effet peut être obtenu par le choix "Trace" du menu "Basic" de DragsterEdit.

•*IC instruction* **TROFF**

--- effet visible seulement dans l'interprèteur

Arrêt du mode espionnage

Syntaxe: **TROFF**

Voir aussi: **TRON**

NOTE: Le même effet peut être obtenu par le choix "Trace" du menu "Basic" de Dragster Edit.

•*IV instruction* **UNDERLINE**

Permet de passer en mode "souligné" sur le minitel distant

Syntaxe: **UNDERLINE** expression_numérique

ex: **UNDERLINE** 0: **REM** arrêt du mode souligné
UNDERLINE 1: **REM** mode souligné

•*IF* instruction* **UNLOCK**

Spécifique à la version compilée multi-tâches. libère un fichier précédemment "**LOCK**é".

Syntaxe: **UNLOCK** file_number
où file_number est une expression numérique donnant le numéro logique d'un fichier ouvert.

ex: **UNLOCK** 1

Voir aussi: **LOCK**

•*FC fonction chaîne* **UPC$**

Met en majuscules la chaîne donnée en paramètre.

Syntaxe: **UPC$**(chaîne)
où chaîne est une expression chaîne

ex: **UPC$**("BonJOur")
rend "BONJOUR"

Voir aussi: **LWC$**

Rend la valeur numérique de l'expression chaîne donnée en paramètre.

Syntaxe: **VAL**(expression_chaine)

ex: **VAL**("123") rend 123

voir aussi: **STR$**

Permet de saisir les zones précédemment déclarées avec l'instruction **ZONE**.

Toutes les zones sont affichées avec leurs valeurs par défaut (contenues dans les variables) et complétées avec des ".". Le curseur est ensuite positionné sur la zone de votre choix (paramètre du **WAIT**).
On sort du **WAIT** dans les cas suivants:

- Time out ou déconnexion, **KEY** rendra 1
- frappe de la touche REPETITION, **KEY** rendra 2, vous devez alors redessiner l'écran et revenir en saisie
- frappe de la touche ENVOI, **KEY** rendra 3
- frappe de la touche GUIDE, **KEY** rendra 4
- frappe de la touche SOMMAIRE, **KEY** rendra 5

Les touches RETOUR et SUITE permettent de changer de zone en cours de saisie (géré par le **WAIT**). Cependant, s'il n'y a qu'une seule zone de saisie, **WAIT** sera arrêté et **KEY** rendra respectivement 6 et 7 pour SUITE et RETOUR.

Les touches CORRECTION et ANNULATION permettent de corriger l'édition d'une zone: CORRECTION supprime le dernier caractère entré et ANNULATION supprime la zone entière (géré par le **WAIT**). Cependant, si la zone courante est de longueur nulle, **WAIT** sera arrêté et **KEY** rendra respectivement 8 et 9 pour ANNULATION et CORRECTION.

La fonction **STARFLAG** vous indiquera la présence d'une étoile en première position de la zone de saisie. Consultez les recommandations aux partenaires Télétel (annexe 1) pour avoir un comportement conforme à ces recommandations.

Syntaxe: **WAIT** zone_number
où zone_number est une expression numérique.

ex:
```
SAISIE:
      WAIT 1: REM saisie des zones avec posit. sur la 1ere
      ON KEY GOTO XTIME,DESSIN,ENVOI,GUIDE,SOMM
      GOTO SAISIE
```

Voir aussi: **KEY**, **STARFLAG**, **ZONENUMBER**

•IV instruction **WAITCONNECT**

Attente d'une connexion. Le programme est arrêté tant que la voie est déconnecté.

Syntaxe: **WAITCONNECT** MaxSecs
où MaxSecs est le nombre de secondes à attendre
une connexion.La valeur 0 indique une attente infinie

Voir aussi: **DISCONNECT**, **CONNECTED**, **STATUS$**

•IC instruction **WHILE-WEND**

Permet d'exécuter des boucles à l'aide d'une expression. A chaque **WHILE** doit correspondre un **WEND** unique qui délimite la fin de la boucle. Les **WHILE** peuvent être imbriqués.

Syntaxe: **WHILE** expression_numérique
instructions
WEND
La boucle sera exécutée tant que l'expression numérique du **WHILE** sera non nulle ou vraie.

ex:
```
X = 9
WHILE X: REM tant que X est différent de 0
      PRINT X
      X = X - 3
WEND
```

aura pour résultat:
9
6
3

WHILE X est équivalent à **WHILE** X<>0

Voir aussi: **BREAK**, **CONTINUE**, **FOR-NEXT**

Permet d'écrire dans un fichier ouvert à la position courante

Syntaxe: **WRITE** file_number,expression,...

ex: **WRITE** 1,A,NOM$,C$
écrit les variables A, NOM$ et C$ dans le fichier 1

Voir aussi: **READ**, **FPRINT**

NOTE: un séparateur de champ TAB est écrit entre deux expressions et un séparateur CR est écrit à la fin du WRITE

•*IC instruction* **YIELDCPU**

Spécifique à la version compilée multi-tâches. Permet de "donner la main" aux autres tâches dans le cas de boucles importantes. Si vous avez à faire une boucle importante sans instructions d'entrée/sortie minitel ou fichier, il est judicieux de placer une instruction YIELDCPU pour ne pas bloquer le noyau multi-tâche de Dragster.

Syntaxe: **YIELDCPU**

ex:
```
FOR I = 1 TO 100000
      GOSUB MaRoutine: REM calcul xxyy
      YIELDCPU: REM on donne la main au système
NEXT
```

Déclaration d'une zone de saisie pour le **WAIT**. Cette déclaration ajoute une zone aux zones précédemment déclarées. Pour remettre à zéro le nombre de zones, utiliser l'instruction **RESETZONES**.

Syntaxe: **ZONE** ligne,colonne,longueur_max,variable,attributs
où ligne, colonne, longueur_max et attributs sont des expressions numériques.

ligne et **colonne** définissent la position de la zone sur l'écran du Minitel (voir **LOCATE**),

longueur_max indique le nombre maximum de caractères qui peuvent être saisis dans cette zone (de 0 à 240)

attributs défini les attributs de la zone:

128:	double largeur
64:	double hauteur
32:	inversion
16:	clignotement
8:	curseur invisible
0 à 7:	couleur des caractères de la zone

Ces valeurs sont cumulables: si attributs = 103, (103 = 64 + 32 + 7) la zone est en double hauteur, inversée et de couleur blanche. En fin de saisie, le résultat de la saisie se retrouve dans la variable.

ex: **ZONE** 10,12,8,A$,7

TRANSPAC

Attention Transpac ne peut pas gérer certains de ces attributs.

Rend le numéro de zone courante au moment de la sortie d'un **WAIT**.

Syntaxe: **ZONENUMBER**

ex: A = **ZONENUMBER**

Voir aussi: **ZONE**, **WAIT**

Annexes

Table 1: Correspondance des couleurs

Valeur	Couleur	Luminosité
0	Noir	0 %
1	Bleu	40 %
2	Rouge	50 %
3	Magenta	60 %
4	Vert	70 %
5	Cyan	80 %
6	Jaune	90 %
7	Blanc	100 %

Table 2: Table ASCII

Valeur	Caractère	Valeur	Caractère	Valeur	Caractère	Valeur	Caractère
0	NUL	32		64	@	96	`
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	'	71	G	103	g
8	BS	40	(	72	H	104	h
9	HT	41	)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DC1	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	ETB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESCAPE	59	;	91	[	123	{
28	FS	60	<	92	\	124	l
29	GS	61	=	93	]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	_	127	DEL

Table 3: Codes d'erreur du Basic Dragster

Ces valeurs de code d'erreur sont rendus par la fonction **ERROR** du Basic de Dragster ou en cours d'analyse de syntaxe ou de compilation.

Code	Signification
0	Pas d'erreur
1	Erreur de syntaxe
2	Mauvaise valeur de paramètre
3	Etiquette utilisée et non définie
4	Redéfinition d'une étiquette ou d'une variable
5	Mauvais type de paramètre
6	Mauvais nombre de paramètres
7	For sans Next
8	Next sans For
9	While sans Wend
10	Wend sans While
11	Division par zéro
12	Return sans Gosub
13	Code reservé n° 1
14	Mauvais appel à une fonction
15	conflit de types
16	trop d'imbrications de While/For/If
17	trop d'imbrications de While/For/If
18	Else sans If
19	Endif sans If
20	Break sans While/For
21	Continue sans While/For
22	Fichier non trouvé ou non interprétable (pas analysé)
23	Double définition d'étiquette
24	Mauvais numéro de fichier
25	Fichier non ouvert
26	Fichier déjà ouvert
27	Indice de tableau erroné ou tableau non déclaré
28	fonction ou procédure non dispo en interprété

Des codes d'erreurs rendus par le système d'exploitation du Macintosh peuvent aussi être rendus par la fonction **ERROR**: les codes indiqués en gras sont généralement des erreurs graves qui impliquent un mauvais état des disques ou disquettes installés. D'autres valeurs peuvent être retournées, consultez Inside Macintosh.

Code	**Signification**
-33	Répertoire plein: vous ne pouvez pas ajouter de fichiers
-34	Disque plein
-35	Pas de volume correspondant
-36	Erreur d'entrée/sortie - grave
-37	Mauvais nom de fichier ou de volume
-38	Fichier non ouvert - ne doit jamais arriver
-39	Fin de fichier atteinte pendant la lecture
-40	Tentative de positionnement avant le début du fichier
-42	Trop de fichiers ouverts - ne doit jamais arriver
-43	Fichier non trouvé
-44	Disquette protégée
-45	Fichier verrouillé
-46	Volume verrouillé
-47	Fichier en cours d'utilisation (**KILL**)
-48	Nom existant déjà (**RENAME**)
-53	La disquette a été éjectée
-52	erreur du Système pendant un **FPOS**
-59	erreur du Système pendant un **RENAME**
-120	Chemin d’accès incorrect (nom de dossier incorrect ou dossier inexistant)

Enfin, d'autres code sont rendus par la fonction **ERROR** et **KEY** lors des saisies sur le minitel connecté avec **WAIT** et **INPUT**:

ERROR	**KEY**	**Signification**
128	1	Time Out ou déconnexion
129	2	Frappe de la touche "Répétition"
130	3	Frappe de la touche "Envoi"
131	4	Frappe de la touche "Guide"
132	5	Frappe de la touche "Sommaire"
133	6	Frappe de la touche "Suite"
134	7	Frappe de la touche "Retour"
135	8	Frappe de la touche "Annulation"
136	9	Frappe de la touche "Correction"

Mots réservés du Basic Dragster (par ordre alphabétique)

ABS
AND
APPEND
ASC
BACKCOLOR
BREAD
BREAK
BWRITE
CANBLOCK
CANEOL
CHR$
CLOSE
CLS
CONNECTED
CONTINUE
CONTROLSN
CREATE
CURPOS$
CURSOR
DATE$
DATE2SECS
DECSCREEN
DELAY
DELSPCL$
DELSPCR$
DEQUEUE
DIAL
DIM
DISCONNECT
DRAW
DRAWSCREEN
ECHO
END
ENQUEUE
EOF
ERROR
FLASH
FONT
FOR
FORECOLOR
FORMAT$
FPOS
FPRINT
FRONTSCREEN
GET$
GETEOF
GETFILE$
GETFINFO
GETID$
GETPAQ$
GETPRIORITY
GETVOL$
GOSUB
GOSUBSCREEN
GOTO
GOTOSCREEN
IF
INPUT
INSTR
INVERSE
KEY
KILL
LEFT$
LEN
LOADSCREEN
LOCATE
LOCK
LOGTIME
LOWER
LWC$
MESSAGE
MESSDOWNLOAD$
MESSUPLOAD
MID$
MOD
MODNUMBER
NEWFOLDER
NEXT
NOT
ON
OPEN
OR
PEND
POST
PRINT
PRINTSCREEN
QUEUESIZE
READ
REM
RENAME
REQUEST
RESETQUEUE
RESETZONES
RESTART
RETURN
RIGHT$
RLEN
RSEEK
SCROLL
SECS
SECS2DATE$
SEEK
SERCONFIG
SETEOF
SETID
SETINFO
SETPRIORITY
SHARED
SIZE
SPC$
STARFLAG
STATUS$
STR$
STRING$
SWMODEM
SYSPARM
TASKNUMBER
TICKCOUNT
TIME$
TIMEOUT
TRACE
TROFF
TRON
UNDERLINE
UNLOCK
UPC$
VAL
WAIT
WAITCONNECT
WEND
WHILE
WRITE
YIELDCPU
ZONE
ZONENUMBER

Mots réservés du Basic Dragster (par catégorie)

Fichiers:

APPEND
BREAD
BWRITE
CLOSE
CREATE
EOF
ERROR
FPOS
FPRINT
GETEOF
GETFILE$
GETFINFO
GETVOL$
KILL
LOCK
NEWFOLDER
OPEN
READ
RENAME
RLEN
RSEEK
SEEK
SETEOF
SETINFO
UNLOCK
WRITE

Types de données:

ASC
CHR$
STR$
VAL

Présentation / affichage:

BACKCOLOR
CANBLOCK
CANEOL
CLS
CURPOS$
CURSOR
DRAW
DRAWSCREEN
ECHO
FLASH
FONT
FORECOLOR
INVERSE
LOADSCREEN
LOCATE
MESSAGE
PRINT
PRINTSCREEN
SCROLL
SIZE
UNDERLINE

Calcul:

ABS
MOD

Logique:

AND
NOT
OR

Gestion des Saisie:

GET$
INPUT
KEY
LOWER
RESETZONES
STARFLAG
WAIT
ZONE
ZONENUMBER

Instructions de contrôle:

BREAK
CONTINUE
END
FOR
GOSUB
GOSUBSCREEN
GOTO
GOTOSCREEN
IF
NEXT
ON
RETURN
WEND
WHILE

Contrôle de la communication:

CONNECTED
DIAL
DISCONNECT
FRONTSCREEN
GET$
GETID$
GETPAQ$
LOGTIME
MESSDOWNLOAD$
MESSUPLOAD
MODNUMBER
SETID
SWMODEM
SYSPARM
STATUS$
TIMEOUT
WAITCONNECT

Dates, temps, heure:

DATE$
DATE2SECS
LOGTIME
SECS
SECS2DATE$
TICKCOUNT
TIME$

Déclarations:

DECSCREEN
DIM
SHARED

Multi-tâche:

DELAY
DEQUEUE
ENQUEUE
GETPRIORITY
PEND
POST
QUEUESIZE
REQUEST
RESETQUEUE
SHARED
TASKNUMBER
YIELDCPU

Traitement des chaines:

CHR$
DELSPCL$
DELSPCR$
FORMAT$
INSTR
LEFT$
LEN
LWC$
MID$
RIGHT$
SPC$
STR$
STRING$
UPC$

Aide au debugging:

TRACE
TROFF
TRON

Divers:

CONTROLSN
ERROR
REM
RESTART
SERCONFIG

Recommandations aux partenaires Télétel

Ces recommandations sont tirées du document suivant:

Recommandations aux partenaires Télétel
Utilisation des touches de fonctions du Minitel

Ces recommandations ont pour but d'uniformiser l'interface utilisateur des serveurs Vidéotex pour Minitel. Vous êtes libres de ne pas vous plier à ces recommandations, mais ce sont quand même des recommandations...

Ces recommandations distinguent deux phases différentes:

- Saisie d'un formulaire (plusieurs zones de saisie à l'écran)
- réponse (une seule zone de saisie ou aucune zone de saisie)

Touche de fonction	Phase de saisie d'un formulaire	Phase de réponse du service
Envoi	Validation d'une ou plusieurs chaînes de caractères, sans obligation de remplir les champs.	
Répétition	Réaffichage de l'écran avec les informations déjà saisies	Réaffichage de l'écran tel qu'il avait été transmis par le serveur
*Répétition	Pas de signification	Réaffichage de l'écran avec prise en compte des mise à jour de la base de données intervenues depuis la transmission précédente
Sommaire	Accès au sommaire de la partie du service en cours de consultation	
*Sommaire	Accès au sommaire du plus haut niveau dans le cas de sommaires hiérarchiques	
Guide	Assistance à la consultation	
*Guide	Appel d'opérateur d'assistance (non dispo)	
Correction	Effacement du caractère précédant le caractère courant. Si le champ courant est vide ou en absence de champ:	
	Pas de signification	Retour à l'écran de la demande avec conservation des informations antérieures et possibilité de les corriger

Touche de fonction	Phase de saisie d'un formulaire	Phase de réponse du service
Annulation	Effacement du champ courant. Si le champ courant est vide ou en absence de champ:	
	Pas de signification	Annulation de la demande et possibilité d'en faire une autre par retour à la phase initiale de saisie
*Annulation	Annulation de tous les champs du formulaire et curseur positionné sur le premier caractère du champ	Pas de signification
Suite	Champ suivant	Page suivante
Retour	Champ précédent	Page précédente
*Suite	Page suivante	Document suivant
*Retour	Retour sur le premier champ de la page précédente. S'il n'y a pas de page précédente, retour au premier champ de la page courante.	Retour au dernier choix

Références Techniques

STUM (Spécification Techniques d'Utilisation du Minitel)
(Document de référence concernant le Minitel)

CNET Paris A
Département Documentation Technique
38/40, avenue du Général Leclerc
92131 Issy les Moulineaux
Tél: (1) 45 29 61 37

Recommandations aux partenaires Télétel
(Décrit l'utilisation recommandée des touches de fonction du Minitel)

Fourni avec les STUM.

Particularités de programmation de Dragster Télétel

Ces modifications ne concernent pas les adaptateurs de type "NAMTEL".

Modification de STATUS$

La fonction `STATUS$` du BASIC Dragster a été modifiée pour fonctionner avec les nouveaux boîtiers Transpac comme suit:

Syntaxe: `STATUS$`

Cette fonction renvoie une chaîne qui a le format suivant:

1er car:	'M' pour toutes les versions actuelles
	'T' indique l'ancienne version Transpac de Dragster
2ème car:	'C' si connecté
	'F' si déconnecté
3ème car:	'0' si serveur, autre si service local du modem
4ème car:	'L' si modem Dragster en mode "Local"
	'R' si modem Dragster en mode "Réseau téléphonique"
	'T' si Dragster Télétel
5ème car:	Phase du modem (prog. avancée)
6ème car:	Dernière cause d'erreur du modem (prog. avancée)
7ème car:	'A' si mode normal
	'R' si modem inversé (retourné avec SWMODEM)

La lecture du `STATUS$` complet dans le cas d'un boîtier DRAGSTER Télétel (reconnaissable au 4ème caractère du `STATUS$`) se fait par l'instruction '`MESSDOWNLOAD$(0)`'.

Exemple:

```
IF MID$(STATUS$,4,1)="T"
  S$=MESSDOWNLOAD$(0)
ELSE
  S$=STATUS$
ENDIF
```

`MESSDOWNLOAD$(0)` renvoie une chaîne qui a le format suivant:

1er car:	M
2e car:	F si voie non connectée
	C si voie connectée
3-7 car:	X (non utilisés)
8-18 car:	numéro de l'appelant (PAV ou serveur)
	à noter:
	le 9e et 10e car donnent le département
	où se trouve le PAV ou serveur appelant
	le 17e et 18e donnent la sous adresse d'appel
	fournie par Télétel pour reconnaître le type d'accès
	utilisé (liste ci-dessous)
19 car:	*mode de taxation (0 si Modem RTC)*
	1 si taxation au demandé (3613)
	0 si taxation au demandeur (3614/3615)
20-21 car:	*groupe fermé d'abonnés (GFA)*
	00 si GFA télétel 2 (3614)
	01 si GFA télétel 3 (3615)
22-33 car:	données d'appel ou mot de passe, cadré à gauche,
	et complété par des espaces.

(si Modem RTC:"MODEM R.T.C.N°:xxx")
Les données sont valables tant que la voie est connectée.

Remarque: Les informations en *italique* sont obsolètes.

Les sous adresses d'appel fournies par Transpac sont les suivantes:

Numéro d'appel	Sous adresse	Taxation Télétel	
3605	80	(t00)	
3613	81	(t01)	
3621	81	(t01)	
3614	82	(t20)	
3615	97	(t32)	palier bas
3615	83	(t34)	palier normal 3615
3615	70	(t36)	palier haut
3616	92	(t34)	palier normal
3616	84	(t36)	palier haut
3617	85	(t44)	
3627	85	(t44)	
3628	90	(t60)	
3629	91	(t70)	

Exemple: 3615 palier normal: sous adresse d'appel: 83

(Ces informations proviennent de la lettre de Télétel N°15)

Modification de WAITCONNECT

Pour éviter les problèmes dûs à des connexions et déconnexions rapides, l'instruction `WAITCONNECT` de Dragster a été modifiée. Voici à quoi correspondent ce que nous appelons les “problèmes de connexion et déconnexion rapides”:

* Un utilisateur A appel le serveur et arrive en voie 0.
* Il se déconnecte.
* Un utilisateur B se connecte juste après et arrive donc sur la voie 0.

Si Dragster était occupé pendant ce court laps de temps, votre logiciel ne s’apercevra pas que l’utilisateur a changé. Le problème peut être grave si les utilisateurs A et B ne désiraient pas accéder aux même services !

La solution à ce problème est la suivante:

• Utiliser une ROM spéciale (actuellement version 25a)
• Changer le `WAITCONNECT` de votre programme par ce qui suit:

```
CONNECT:
  X$=STATUS$
  WAITCONNECT 10
  IF NOT CONNECTED:GOTO CONNECT:ENDIF
```

Lorsque l’utilisateur A se déconnecte, Dragster restera en état “virtuellement déconnecté” jusqu’au prochain `STATUS$` qui le remettra en état réel (connecté ou non). Tant que Dragster sera en état “virtuellement déconnecté” l’utilisateur B restera en attente au niveau du boîtier Dragster Télétel, sur le message “Système serveur Dragster X25...”.

Index

A