MASQUE : UN GÉNÉRATEUR/GESTIONNAIRE D'ÉCRANS UTILISABLE PAR DIVERS LANGAGES DE PROGRAMMATION.

Gérard Cochonneau
(Informatique-Hydrologie)
(UR 502, Cadres Spatiaux de l'Indépendance Alimentaire)

EMBRAPA
Empresa Brasileira de Pesquisa Agropecuária
SuperCenter Venâncio 2.000
70333 - Brasília, DF (Brésil)

RESUME - Le caractère interactif des matériels informatiques disponibles depuis déjà de nombreuses années a conduit tout naturellement à développer des programmes qui communiquent avec l'opérateur, soit sous forme de questions-réponses qui se succèdent sur l'écran, soit à l'aide de grilles d'écran. Cette seconde technique, beaucoup plus performante et attrayante pour l'utilisateur apparaît vite indispensable pour la consultation ou la manipulation de données, même peu volumineuses. En contrepartie et compte tenu des ressources limitées offertes dans ce domaine par les langages de programmation, elle est assez difficile à mettre en oeuvre sans l'utilisation d'un outil spécifique.

Cette note présente un outil opérationnel sur micro-ordinateur compatible IBM-PC, qui permet :

- d'une part, de définir ou modifier des grilles d'écran ;
- d'autre part, d'utiliser ces grilles d'écran à l'intérieur de programmes pour saisir ou contrôler des données.

Conçu pour répondre aux besoins identifiés lors d'une expérience précédente sur mini-ordinateur, ce gestionnaire d'écran a d'abord été développé en langage évolué (Cobol). Les résultats satisfaisants obtenus, lors de l'utilisation de cette première version pour le développement du logiciel Hydrom, ont conduit à la mise au point d'une seconde version écrite en Assembleur Conditionnel, donc plus performante et utilisable par différents langages de programmation (Cobol, Basic compilé, Quick-Basic, Fortran-77).

L'éditeur permet de définir, consulter, imprimer les grilles d'écran ; à tout moment, lors de la définition, la grille peut être montrée à l'écran dans son état partiel. On y distingue des champs constants et des champs variables.

Le contenu des champs constants est fixé lors de la définition et chaque champ variable est caractérisé par des attributs obligatoires :

- son nom qui suffira pour l'identifier dans la grille ;
- sa position sur l'écran ;
- sa longueur ;
- le type de données qu'il accepte (alphanumérique, numérique signé ou non signé, alphabétique, etc.) ;

et un certain nombre d'attributs optionnels pour indiquer :

- que le champ est obligatoire lors de la saisie ;
- que ce champ devra être totalement rempli ;
- une valeur initiale qui sera affichée automatiquement lors de l'initialisation ;
- des valeurs extrêmes qui ne peuvent être dépassées lors de la saisie.

A l'intérieur d'un programme, l'utilisation des grilles se fait par l'appel d'un module capable d'effectuer à la demande cinq fonctions principales :

- affichage d'une grille prédéfinie ;
- récupération de l'information saisie par l'opérateur dans un champ ou un groupe de champs variables ;
- écriture dans un champ ou un groupe de champs variables ;
- réinitialisation de la grille ou d'un groupe de champs variables ;
- émission d'un message d'erreur fourni par le programme utilisateur.

Lors de la saisie, l'opérateur peut, au lieu de remplir un champ, actionner une touche de fonction ; ce qui sera communiqué au programme principal à travers un code de retour et pourra être utilisé dans le programme principal pour déclencher certaines opérations particulières (ruptures de séquence, répétition du contenu d'un champ, etc.).

INTRODUCTION

Le caractère interactif des matériels informatiques disponibles depuis déjà de nombreuses années a conduit tout naturellement à développer des programmes qui communiquent avec l'opérateur, soit sous forme de questions-réponses qui se succèdent sur l'écran, soit à l'aide de grilles d'écran. Cette seconde technique, beaucoup plus performante et attrayante pour l'utilisateur final apparaît vite indispensable pour la consultation ou la manipulation de données, même peu volumineuses. En contrepartie, et compte tenu des ressources limitées offertes dans ce domaine par les lan-

gages de programmation, elle est assez difficile à mettre en oeuvre sans l'utilisation d'un outil spécifique.

Mis au point et utilisé à l'ORSTOM, Masque est un logiciel de génération et de gestion de grilles d'écran qui répond à ces besoins. A l'heure actuelle, il est opérationnel pour les langages de programmation suivants :

- Basic Compilé de MicroSoft ;
- Quick Basic de MicroSoft ;
- Level-II Cobol de MicroFocus ;
- Fortran-77 de MicroSoft ;
- TurboPascal de Borland.

1. DESCRIPTION GENERALE

Une grille d'écran se présente à l'utilisateur comme un partage de l'écran en zones constantes dont le contenu est prédéfini et en zones variables réservées à l'entrée de données qui sont alors récupérées par le programme d'application qui utilise la grille d'écran. Cette idée de *"récupération"* des données est d'ailleurs souvent artificielle, dans la mesure où, généralement, l'application ne lit pas les données dans la zone variable mais les mémorise au fur et à mesure de leur digitation et se sert parallèlement de la zone variable pour informer l'utilisateur de ce qu'il a digité. Une grille est donc comparable à un formulaire comme on en rencontre dans la vie courante (déclaration d'impôts, fiche de renseignements etc...). A ce propos, lorsqu'un système utilise des grilles d'écran pour saisir des formulaires préalablement remplis, il est très important de concevoir des grilles qui ressemblent le plus possible aux formulaires, afin de faciliter les opérations de saisie et de minimiser les risques de confusion ou d'oubli d'informations. On emploie également le terme *"masque d'écran"* au lieu de grille, l'ensemble des champs constants pouvant être vu comme un masque collé sur l'écran, la partie dynamique de l'écran se réduisant alors aux champs variables (les trous du masque). Nous ne traiterons pas séparément les menus qui seront considérés comme un cas particulier de grille avec un seul champ variable (le numéro de l'option à choisir) et autant de champs constants que d'options offertes. Des menus plus professionnels peuvent être élaborés en définissant un champ variable devant chaque option et en déplaçant une flèche d'un champ à l'autre pour faire son choix.

Le logiciel Masque se compose de deux parties distinctes et indépendantes (Figure 1) :

- un générateur de grilles d'écran qui permet de définir des grilles, de les modifier, de les imprimer, de les tester ;

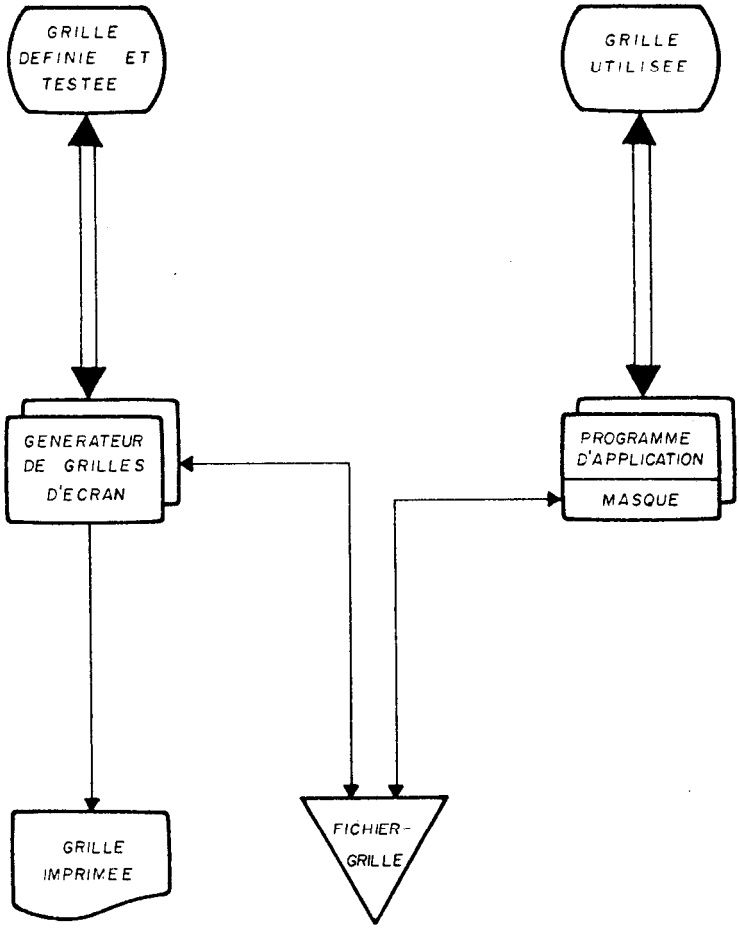
GRILLE DEFINIE ET TESTEE
GRILLE UTILISEE
GENERATEUR DE GRILLES D'ECRAN
PROGRAMME D'APPLICATION
MASQUE
GRILLE IMPRIMEE
FICHIER-GRILLE

Figure 1

- un module appelé Masque qui, intégré dans un programme d'application, permet d'utiliser ces grilles.

Pour le générateur de grilles et pour le programme d'application, une grille est avant tout un fichier que nous conviendrons d'appeler dans toute la suite : fichier-grille. Ce fichier est indépendant du langage de programmation dans lequel sera écrit le programme qui l'utilisera.

Pour le module Masque, une grille est un ensemble d'informations lues dans un fichier, et qui sont nécessaires et suffisantes pour décrire les zones constantes et surtout les champs variables de la grille.

2. DESCRIPTION DES GRILLES D'ECRAN

Un *champ constant* est bien entendu le plus simple : il est parfaitement défini par son contenu et ses coordonnées sur l'écran, exprimées en numéro-de-ligne,numéro-de-colonne. Il peut représenter un titre, des informations destinées à l'utilisateur mais plus généralement, il indique la signification d'un champ variable auquel il est associé.

Un *champ variable* est caractérisé par plusieurs paramètres, obligatoires ou optionnels et identifié par un nom de un à trois caractères qui est utilisé par le programme d'application pour communiquer avec la grille.

L'utilisation d'un nom pour identifier le champ présente plusieurs avantages. Tout d'abord, cela clarifie les relations entre le programme d'application et la grille, si l'on prend soin de nommer les champs en rapport avec leur signification. D'autre part, l'ajout ou la suppression d'un champ sur la grille se traduit de manière relativement souple pour le programme d'application : il suffit d'y ajouter ou d'y supprimer le traitement du champ en question sans intervenir sur les autres champs ; il en est de même dans le cas d'un changement de position du champ sur la grille. Ce ne serait pas le cas si les champs étaient simplement identifiés par un numéro d'ordre lié à leur position sur l'écran, par exemple.

La description du champ variable détermine en fait le type de données qu'il acceptera lors de la saisie. Notons que cette description n'est pas globale mais qu'à chaque position du champ est associé un type de données ; ce qui permet de mélanger, à l'intérieur d'un même champ, des types différents tout en bénéficiant d'un contrôle maximum lors de la saisie. Prenons par exemple le cas d'un champ de huit positions dont les quatre premières doivent être des lettres et les quatre dernières, des chiffres : une description globale obligerait à définir un champ alphanumérique et à contrôler la valeur saisie dans le programme

d'application ; alors qu'une description position par position permet au module Masque d'effectuer un contrôle immédiat et d'interdire l'entrée d'une lettre dans la partie numérique et inversement. Les types de données possibles sont les suivants :

- A : type alphabétique (majuscules ou minuscules non accentuées) ;
- X : type alphanumérique (chiffres de 0 à 9, majuscules, minuscules accentuées ou non) ;
- D : type numérique (chiffres de 0 à 9) ;
- S : type signe (+ ou -) ;
- E : type ASCII étendu (n'importe quel caractère dont le code est supérieur ou égal à 31 dans la table ASCII).

Tout autre caractère utilisé dans la définition du champ sera considéré comme un délimiteur et affiché automatiquement sur l'écran mais, comme un champ constant, il ne sera pas accessible à l'utilisateur lors de la saisie. Les caractères "," et "." sont des délimiteurs particuliers utiles pour séparer les parties entières et décimales des champs en virgule fixe.

Une facilité supplémentaire, semblable à ce qui existe dans la clause Picture du Cobol, est offerte pour répéter le même type de données, ou le même délimiteur, sur des positions consécutives. La description du champ pris comme exemple plus haut peut s'écrire indifféremment AAAADDDD ou A(4)D(4).

La combinaison de ces types de données et des délimiteurs permet ainsi de définir des champs simples (numériques, alphabétiques...) ou des champs composés (dates, téléphones, numérique signé avec ou sans point décimal etc...).

Le dernier paramètre obligatoire pour définir pleinement un champ variable est la position à laquelle il commence sur l'écran exprimée, comme pour les champs constants, par un numéro de ligne et un numéro de colonne.

Remarquons que la longueur du champ n'est pas un paramètre nécessaire puisqu'elle est implicitement définie par la description position par position du type de données. Toutefois, un champ ne peut avoir une longueur totale supérieure à 80 et doit se trouver entièrement sur la même ligne.

Pour compléter la description d'un champ variable, des paramètres optionnels sont disponibles :

- un attribut qui rend le champ obligatoire lors de l'utilisation de la grille, l'opérateur ne pourra alors passer ce champ sans y introduire une valeur en accord avec le type de données qu'il peut recevoir ;
- un attribut qui, lors de l'utilisation de la grille, interdira à l'opérateur de passer tant que le champ ne sera pas rempli totalement ;

- une valeur initiale avec laquelle le champ sera rempli dès l'affichage de la grille (cette valeur sera bien sûr modifiable par l'opérateur lors de la saisie ;

- un intervalle des valeurs que peut prendre le champ, défini par une valeur minimale et une valeur maximale ; ce paramètre introduit un contrôle supplémentaire au niveau du module Masque, contrôle qui est ainsi économisé dans le programme d'application.

Ces valeurs minimales et maximales ne sont autorisées que si la longueur totale du champ ne dépasse pas cinq. Malgré ce facteur limitant, ces paramètres sont intéressants pour des champs qui correspondent à un numéro de mois, à une option, etc..

Si le nombre et la taille des champs fixes ne sont limités que par la taille de l'écran (24 lignes de 80 colonnes), un certain nombre de limites en relation avec les champs variables sont imposées par le logiciel. C'est ainsi que, pour une même grille le nombre de champs doit rester inférieur à 220 et le total des zones variables ne peut dépasser 1200 positions soit 62% de la surface totale de l'écran. Ces valeurs maximales ont été choisies suffisamment élevées pour ne pas représenter une contrainte très sévère.

3. LE GENERATEUR D'ECRANS

Ecrit en Turbo-Pascal, le générateur de grilles d'écran se présente comme un module exécutable capable de réaliser cinq fonctions que nous allons décrire dans ce chapitre. Il utilise lui-même le module Masque pour communiquer avec l'utilisateur. Il est indispensable, lors du développement de l'application, pour définir les grilles, mais il est inutile de le fournir à l'utilisateur final.

3.1. Visualiser une grille

Il s'agit tout simplement de faire apparaître sur l'écran une grille déjà enregistrée dans un fichier-grille. Cette présentation de la grille est identique à ce qu'elle sera lors de son utilisation dans un programme d'application.

3.2. Manipuler une grille

C'est la fonction la plus importante offerte par le générateur puisqu'elle permet de définir et corriger de manière interactive des grilles d'écran avant de les conserver dans des fichier-grilles. Pour décrire chaque champ variable, l'utilisateur doit remplir une grille d'écran avec tous les paramètres obligatoires et éventuellement des paramètres optionnels. Pour décrire les champs constants, deux manières peuvent être employées : procéder comme pour les champs variables ou définir le masque directement en plaçant le

contenu des zones constantes à leur position voulue sur un écran vierge. Les mêmes facilités sont disponibles pour corriger ou supprimer un champ. Le logiciel effectue au fur et à mesure les contrôles nécessaires pour vérifier que la description est conforme aux règles exposées plus haut. Enfin, à tout moment, l'utilisateur peut afficher sur l'écran l'état de la grille en cours de définition et la sauver dans un fichier-grille avec possibilité d'en changer le nom ou le répertoire.

3.3. Imprimer une grille

La figure 2 montre un exemple de listing produit par cette fonction. Le logiciel lit la grille à imprimer dans un fichier-grille et produit une impression en deux parties : la grille telle qu'elle sera affichée, placée entre deux gabarits de numéros de colonnes et un tableau donnant les caractéristiques des champs variables. Pour chacun des champs variables, sont consignés dans ce tableau, le nom du champ, sa position, sa description, ses attributs (M : champ obligatoire, T : champ à remplir complètement, I : champ avec valeur initiale), ses valeurs initiales, minimales et maximales.

Ce listing peut être archivé ou utilisé pour contrôler la définition ou comme point de départ pour définir une autre grille, mais il est aussi indispensable lors de la définition du programme qui va utiliser la grille.

3.4. Tester une grille

L'utilisateur peut profiter de cette fonction pour simuler un emploi réel de la grille qu'il a défini et sauvegardé. Cela lui permet, avant même d'écrire son programme d'application, de vérifier la fonctionnalité de la grille et la bonne définition des champs variables (type de données, valeurs initiale, minimale, maximale, longueur de champ). La saisie se fait dans l'ordre dans lequel ont été définis les champs ; cet ordre est en général différent de celui que l'utilisateur s'imposera dans son programme d'application. Les valeurs saisies ne sont bien entendu pas conservées.

3.5. Compiler une grille

Bien que redondante avec la fonction de manipulation de grilles, cette fonction a quand même été développée. Elle transforme une description de grille saisie dans un autre éditeur de texte en fichier-grille compatible avec le logiciel Masque. Un utilisateur occasionnel peut ainsi éviter d'apprendre à se servir de la fonction de manipulation et définir sa grille à l'aide d'un éditeur de texte qui lui est plus familier. Le fichier ainsi saisi, que nous conviendrons de nommer fichier-source doit cependant respecter un format strict décrit dans la notice d'utilisation : un enregistrement pour chaque

```
ORSTOM   - Masque        Définition   de   la   grille   DEMO.GRC
Page 1

 ...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
ORSTOM                                                    LOGICIEL "MASQUE"
                          EXEMPLE D'UTILISATION

 Nom    : ..............................      Né(e) le : ../../....
 Prénom : ..............................      Indice   : ...

 Affectation : MET (MET/O-M)
 Téléphone   : '..§-..-..-..

 Année  : ....                                Aide aux touches de fonction:
 Mois   : ..
 Salaire mensuel brut : ......,..   FF          F1 : retour en arrière
 Salaire mensuel net  : ......,..   FF          F5 : duplication d'un champ
                                                F3 : réinitialisaion
 Validation : ... (OUI/NON)                     SF1: fin de saisie

 ...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
```

ORSTOM - Masque Définition de la grille DEMO.GRC Page 2

NO	Nom	Lig	Col	Description	Attributs	Val.Ini	Val.Min	Val.Max
1	NOM	05	12	X(30)	M			
2	PRE	06	12	X(30)	M			
3	D	05	61	DD/DD/DDDD	T			
4	IND	06	61	DDD	T			
5	AF	08	16	XXX	M T I	MET		
6	TEL	09	16	'DD$-DD-DD-DD	T			
7	A	12	11	DDDD	M T			
8	M	13	11	DD	M T		01	12
9	SB	14	25	SD(5),DD	M			
10	SN	15	25	SD(5),DD	M			
11	VAL	17	15	AAA	M T			

Fig. 2 : exemple d'impression de grille.

zone, constante ou variable, contient tous les paramètres nécessaires à la définition du champ. Dès que le fichier source a été compilé avec succès, un fichier-grille est créé et toutes les fonctions du générateur d'écran peuvent être appliquées.

4. LE MODULE MASQUE

Comme nous l'avons résumé plus haut, ce module est dédié à l'utilisation d'une grille déjà définie, dans un programme d'application. Le programme le voit comme un module externe (subroutine ou procédure, selon le langage) et communique avec lui par des paramètres en nombre variable selon la fonction demandée qui est elle-même un des paramètres. Masque réalise *sept fonctions* et dans tous les cas, au moins deux paramètres sont nécessaires : l'un contient la fonction que l'on demande à Masque de réaliser, l'autre est un code de retour. Ce dernier doit être testé au retour de Masque pour détecter si une erreur s'est produite lors de la réalisation de la fonction demandée et identifier cette erreur. Parmi les erreurs possibles, citons le cas où le fichier-grille n'a pas pu être accédé ou est non conforme et le cas où le nom d'un champ communiqué à Masque ne correspond pas à un champ de la grille. Ces erreurs se produisent le plus souvent lors de la mise au point du programme, mais le code de retour joue un autre rôle très important lorsque le programme est opérationnel comme nous le verrons dans la description de la fonction de saisie. La description de chacune de ces fonctions est illustrée par un exemple, en langage Basic et en langage Cobol, basé sur la grille imprimée en figure 2.

La *fonction Get* doit être exécutée avant toute opération à réaliser sur la grille : elle charge la description de la grille à partir du fichier-grille et l'affiche dans son état initial. Le nom du fichier-grille doit être fourni lors de l'appel au module Masque. Chaque position des champs variables est matérialisée par un point, les délimiteurs sont affichés ainsi que les valeurs initiales des champs initialisés.

Basic :

```
10 GRILLE$ = "DEMO."              'GRILLE:nom du fichier-grille
20 OP$     = SPACE$(2)            'OP    :code de l'opération
30 STAT$   = SPACE$(2)            'STAT  :code de retour
...
40 OP$ = "GT"                     ' GT   :opération Get
50 CALL MASQUE(GRILLE$,STAT$,OP$) 'appel de Masque pour
                                   'afficher la grille
```

Cobol :

```
01  GRILLE PIC X(5) VALUE 'DEMO.'.
01  OP      PIC XX.
01  STAT    PIC XX.
    ...
    MOVE 'GT' TO OP.
    CALL 'MASQUE' USING GRILLE STAT OP.
```

La *fonction Read* est la plus souvent utilisée et la plus sophistiquée : elle autorise l'opérateur à saisir des valeurs dans un ou plusieurs champs variables. La liste des noms des champs à saisir doit être communiquée lors de l'appel de Masque ainsi que l'adresse d'une zone qui contiendra, au retour, les valeurs saisies par l'opérateur. Un effort particulier a été fait pour optimiser les performances de cette fonction. Chaque touche actionnée est contrôlée en accord avec le type de données que peut recevoir la position où se trouve le curseur et toute valeur non conforme est refusée et provoque l'émission d'un signal d'alarme. Les positions qui correspondent à des délimiteurs sont passées automatiquement puisqu'elles sont déjà remplies. Les touches de déplacement du curseur vers la gauche ou vers la droite sont utilisables à l'intérieur d'un champ ainsi que le mode insertion et la touche de suppression de caractères ; les caractères qui, éventuellement, déborderaient du champ en mode insertion disparaissent de l'écran mais restent accessibles si d'autres sont supprimés, à condition bien sûr de rester dans le même champ.

La saisie d'un champ est considérée comme terminée si l'opérateur vient d'informer la dernière position du champ ou si la touche Retour-Chariot a été actionnée et que l'état du champ satisfait les conditions imposées par les attributs de champ obligatoire ou à remplir complètement. Dans le cas d'un champ numérique, le logiciel effectue un cadrage automatique à droite et complète le champ avec des zéros non significatifs mais en tenant compte du signe et du point décimal. Par contre, un champ alphabétique ou alpha-numérique est cadré à gauche.

Le code de retour joue un rôle important dans cette fonction de saisie car si l'opérateur actionne une des 47 touches de fonction autorisées (F1 à F40, Esc, etc..) une valeur est renvoyée dans ce code de retour qui doit être testé dans le programme appelant pour faire réaliser une opération particulière telle que l'appel d'un écran d'aide, le passage à un autre champ, la duplication d'une valeur antérieure dans un champ, la sortie du programme etc..

Basic :

```
100 C.DATE$ = "D."                  `C.DATE:nom du champ à saisir
110 V.DATE$ = SPACE$(10)            `V.DATE:pour recevoir la date
                                    `saisie
...
120 OP$ = "RD"                      ` RD   :opération Read
130 CALL MASQUE(C.DATE$,V.DATE$,STAT$,OP$) `appel de Masque
                                    `pour saisir le champ D
140 IF STAT$ <> "00" THEN traiter_touche_de_fonction.
```

Cobol :

```
01  C-DATE PIC XX VALUE 'D.'.
01  V-DATE PIC X(10).
    ...
    MOVE 'RD' TO OP.
    CALL 'MASQUE' USING C-DATE V-DATE STAT OP.
    IF STAT NOT = ZERO THEN traiter_touche_de_fonction
```

La *fonction Reset* permet au programme d'application de remettre dans leur état initial un ou plusieurs champs variables de la grille ; nous entendons toujours par état initial la matérialisation de chaque position par un point ou l'affichage de la valeur initiale. Cette fonction peut par exemple être déclenchée, à la demande de l'opérateur (avec une touche de fonction), pour effacer le contenu d'un champ. La liste des noms des champs à réinitialiser doit être fournie avant l'appel du module Masque sauf dans le cas particulier où l'on souhaite remettre la grille entière dans son état initial. Le résultat est alors apparemment identique à ce que réalise la fonction Get, mais en réalité aucun accès n'est fait au fichier-grille puisque la description de la grille est déjà présente dans la mémoire.

Basic :

```
200 OP$ = "RS"                      ` RS   :opération Reset
210 CALL MASQUE(C.DATE$,STAT$,OP$) `appel de Masque pour
                                    `réinitialiser le champ D
```

Cobol :

```
MOVE 'RS' TO OP.
CALL 'MASQUE' USING C-DATE STAT OP.
```

L'écriture directe dans un champ, par le programme d'application se fait grâce à la *fonction Write*. La liste des noms des champs que l'on désire remplir ainsi qu'une zone qui contient les valeurs à afficher doivent être communiquées au module Masque lorsqu'il est appelé. Cette opération peut être déclenchée, à la

demande de l'opérateur, pour répéter dans un champ la valeur saisie précédemment mais aussi pour lui communiquer des informations. Son utilisation la plus fréquente correspond à l'affichage de données déjà saisies, en vue de les corriger.

Basic :

```
300 NOM$ = "AF D."                      `nom des champs à aff.
310 VALEUR$ = "O-M25/12/1900"           `affectation+date à afficher
...
320 OP$ = "WR"                          ` WR   :opération Write
330 CALL MASQUE(NOM$,VALEUR$,STAT$,OP$) `appel de Masque pour
                                        `afficher l'affectation et
                                        `la date
```

Cobol :

```
01  NOM PIC X(6) VALUE 'AF D.'.
01  VALEUR.
    02  FILLER PIC X(03) VALUE 'O-M'.
    02  FILLER PIC X(10) VALUE '25/12/1900'.
    ...
    MOVE 'WR' TO OP.
    CALL 'MASQUE' USING NOM VALEUR STAT OP.
```

La *fonction Clear* efface tout simplement l'écran ; elle ne nécessite aucun paramètre particulier.

Basic :

```
400 OP$ = "CL"              ` CL   :opération Clear
410 CALL MASQUE(STAT$,OP$)`appel de Masque pour effacer l'écran
```

Cobol :

```
MOVE 'CL' TO OP.
CALL 'MASQUE' USING STAT OP.
```

La *fonction Display* affiche un message sur la vingt-cinquième ligne de l'écran, provoque l'émission d'un signal d'alarme et attend que l'opérateur appuie sur une touche avant d'effacer le message et de retourner au programme appelant. Le message peut être un message d'erreur ou d'information et doit être communiqué à Masque lors de l'appel.

Basic :

```
500 MSG1$ = "Valeur erronée$"    'message terminé par $
...
510 OP$ = "DP"                    ' DP   :opération Display
520 CALL MASQUE(MSG1$,STAT$,OP$) 'appel de Masque pour afficher
                                  'le message en bas de l'écran
```

Cobol :

```
       77  MSG1 PIC X(15) VALUE 'Valeur erronée$'.
           ...
           MOVE 'DP' TO OP.
           CALL 'MASQUE' USING MSG1 STAT OP.
```

La *fonction Output-Color* permet l'affichage d'une information à partir d'une position quelconque de l'écran et en bénéficiant du choix des attributs d'affichage. La position sur l'écran, les attributs d'affichage (vidéo inversée, clignotement, couleur) ainsi que la chaîne de caractères à afficher doivent être communiqués à Masque lors de l'appel.

Basic :

```
600 MSG2$ = "Calculs en cours$" 'message terminé par $
610 CUR$  = "2015"              'ligne 20, colonne 15
620 COUL$ = "000705"            '07:vidéo inversée
                                '05:clignotement
...
630 OP$ = "OC"                    ' OC   :opération Output-Color
640 CALL MASQUE(MSG2$,CUR$,COUL$,STAT$,OP$) 'appel de Masque
                                  'pour afficher le message
```

Cobol :

```
       77  MSG2 PIC X(17) VALUE 'Calculs en cours$'.
       77  CUR  PIC X(4)  VALUE '2015'.
       77  COUL PIC X(6)  VALUE '000705'.
           ...
           MOVE 'OC' TO OP.
           CALL 'MASQUE' USING MSG2 CUR COUL STAT OP.
```

Notons que ces trois dernières fonctions peuvent être utilisées indépendamment de l'utilisation ou non d'une grille d'écran.

5. HISTORIQUE

L'idée ou plutôt la nécessité de développer ce logiciel de grilles d'écran est apparue à la suite du changement de matériel informatique causé par le transfert du Laboratoire d'Hydrologie. L'utilisation d'un tel outil sur le Mini-6 entre 1981 et 1984 ayant montré tout l'avantage que l'on pouvait en tirer, il s'avérait indispensable de continuer dans cette voie. La première application à adapter sur les micro-ordinateurs IBM-PC étant écrite en Cobol, nous avons cherché un gestionnaire d'écran adapté à nos besoins. A cette date (début 1985), de tels logiciels étaient moins nombreux qu'aujourd'hui et les deux qui furent testés, Display Manager et Forms-II n'étaient pas satisfaisants : l'un n'étant pas utilisable par le langage Cobol, l'autre étant par trop rudimentaire. La décision a donc été prise de développer un outil qui soit assez ressemblant par ses possibilités et son interface avec celui utilisé précédemment, ceci afin de faciliter l'adaptation des programmes déjà opérationnels.

Ecrite en Level-II-Cobol, comme les programmes de l'application à adapter, la première version se composait d'un module d'utilisation des grilles et d'une partie de manipulation des grilles des plus rudimentaire : utilisation d'un éditeur classique pour saisir les grilles et transformation du fichier saisi, dans un format reconnu par le module d'utilisation. Ce prototype a été utilisé pendant plus d'un an et la première version du logiciel Hydrom, diffusée au printemps 1986 à des fins de test, l'utilisait encore.

Il devenait dès lors indispensable d'améliorer les performances, notamment au niveau de la rapidité d'affichage, et d'introduire des fonctionnalités - telles que le cadrage des valeurs numériques, le mode insertion, la suppression - qu'il n'était pas viable de développer dans un langage évolué, toujours pour des raisons de rapidité d'exécution. D'autre part, certaines applications graphiques d'Hydrom étaient développées en Basic et, pour des raisons d'homogénéité, il était intéressant d'y utiliser le même logiciel de grille d'écran. Une nouvelle version du module d'utilisation des grilles a donc été écrite en Assembleur Conditionnel qui représentait la seule alternative pour, à la fois, augmenter l'efficacité et permettre l'interface avec divers langages de programmation.

Ce n'est que plus récemment que des versions du module Masque ont été développées pour le Fortran-77 et le Turbo-Pascal et qu'un générateur d'écrans plus opérationnel a été mis au point pour en arriver à un produit plus complet.

6. POINTS FORTS ET PRINCIPAUX PROBLEMES RENCONTRES

L'hétérogénéité des compilateurs disponibles pour micro-ordinateur représente un obstacle important dès lors qu'un même module écrit en assembleur doit s'interfacer avec différents langages. En effet, la communication entre un programme appelant et un module appelé est réalisée de façon particulière dans chaque compilateur et varie parfois d'une version à l'autre du même compilateur. Force est de constater aussi que, très souvent, la documentation fournie à ce sujet dans les manuels de référence se résume à un paragraphe d'informations assez imprécises et parfois erronées qu'il est indispensable de vérifier soigneusement. Certains compilateurs (Fortran-77, Basic) produisent un module objet qui nécessite une édition de liens alors que d'autres (Level-II-Cobol, Turbo-Pascal) produisent directement un module exécutable. Le module assembleur appelé doit être adapté à chaque cas. La version pour le Turbo-Pascal est la plus complexe car les adresses des variables internes au module doivent être recalculées en temps d'exécution.

Bien que les paramètres qui sont communiqués entre le programme appelant et le module aient été définis le plus universellement possible comme des chaînes de caractères, il a fallu tenir compte de la façon particulière dont est structuré ce type de variables en Basic (longueur, adresse du contenu) et en Turbo-Pascal (longueur, contenu).

Les fonctions d'affichages offertes par le système opérationnel DOS se révélant assez peu efficaces, l'édition sur l'écran se fait en écrivant directement dans la mémoire, du moins si l'on travaille en mode texte. On arrive ainsi à une vitesse d'affichage assez spectaculaire. L'utilisation du module Masque en mode graphique est également possible, mais l'affichage sur l'écran reste alors tributaire de la lenteur des fonctions du DOS.

L'absence d'un code de retour a été envisagée ; dans le cas d'une erreur détectée dans la routine Masque, on aurait pu se contenter de retourner au DOS après avoir signalé l'erreur. Mais, dans le cas du développement d'un programme en Turbo-Pascal par exemple où le test et l'édition d'un programme se réalisent simultanément, les dernières modifications du programme source auraient été perdues. On pouvait aussi imaginer d'imposer des touches de fonction dans le module Masque pour réaliser des fonctions classiques telles que le retour au champ précédent, la réinitialisation du champ etc.. et s'affranchir ainsi du code de retour utilisé dans la fonction Read pour informer le programme appelant de la touche de fonction qui a été actionnée. Mais ceci aurait entraîné la perte de toute la souplesse qui permet à l'utilisateur de

choisir ses propres touches de fonction et de définir l'opération associée.

La combinaison des fonctions Write et Reset du module Masque permet une sorte de paramétrage de la grille en fonction des informations qui y sont saisies. C'est ainsi que des zones variables peuvent être définies dans la grille, blanchies dès l'affichage initial et utilisées ensuite par couples (intitulé de champ, champ lui-même) pour saisir des informations en rapport avec la valeur donnée par l'opérateur à un autre champ.

Si l'on se réfère aux grilles d'écran habituelles, on peut penser que la possibilité de définir 220 champs variables est exagérée. Cependant cette facilité est parfois intéressante, comme ce fut le cas dans une application peu ordinaire du module Masque : son utilisation dans un programme de saisie sur une table à digitaliser. Le clavier, désactivé, était remplacé par un menu dessiné sur la table et la grille ne servait qu'à visualiser les données saisies et à informer l'opérateur. Mais pour accompagner la saisie faite sur le menu, chaque position des zones variables devait elle-même être isolée et représentée par un champ, afin qu'à chaque saisie de coordonnées dans la zone de menu soit affiché le caractère correspondant ou réalisée la fonction demandée (effacement d'un caractère, retour en arrière, etc...). La possibilité offerte par le logiciel de définir des champs variables contigus était également indispensable.

Il n'est pas courant de réaliser à l'ORSTOM ce genre de travail dans la mesure où l'on n'a pas vocation de développer des outils informatiques d'intérêt général mais plutôt des applications. Comme on a pu le voir, la partie générateur d'écrans a, depuis le début, été défavorisée par rapport au module d'utilisation des grilles qui est le plus important pour l'utilisateur final de l'application. De même, le manque de documentation du logiciel, laissée de côté jusqu'au dernier moment en a réduit l'emploi : mises à part quelques utilisations marginales, seules deux applications, Hydrom et Pluviom, en font usage. Il s'agit toutefois d'applications de grande envergure, Hydrom notamment utilise 62 masques d'écran dont 13 menus.