Utilisation de fenêtres VisualBasic dans MapInfo

Un exemple pour les non-initiés

basé sur des documents produits par

Laurent Maumet geo.maumet@free.fr

et **Sébastien Roddier** sebastien.roddier@geo-hyd.com

et retravaillés par **Jacques Paris** jacques@paris-pc-gis.com

Juin 2002

Requiert l'utilisation de VisualBasic 6, MapBasic et.MapInfo

Contenu

A - Présentation du projet

Fonctionnement global

Lorsque l'application MBX est lancée dans MapInfo, elle lance à son tour le programme EXE qui contient la fenêtre VisualBasic.

La fermeture complète des applications MBX et EXE ainsi que de fenêtres MapInfo qui auraient été ouvertes par l'application est faite par un des boutons de la fenêtre VB.

Fonctions spécifiques

Ces fonctions ont été choisies pour montrer la variaété des environnements dans lesquels ce genre d'application est possible.

1 – Affichage d'une carte

Il s'agit d'afficher à partir de la fenêtre VB une carte dans une fenêtre MapInfo. Dans cet exemple, le nom et la position d'une carte sont codés dans le programme même.

2 – Lecture

Il s'agit d'afficher comme un message VB le contenu de 2 variables qui sont définies dans l'environnement MapInfo.

3 – Écriture

Cette fonction affiche un message dans le contexte MapInfo et va changer la définition (contenu) des deux variables MapInfo. On peut se rendre compte du changement en rejouant « Lecture »

4 – Exécution

Cette fonction affiche sous la forme d'un message VB le résultat de l'appel à une fonction définie dans MapBasic

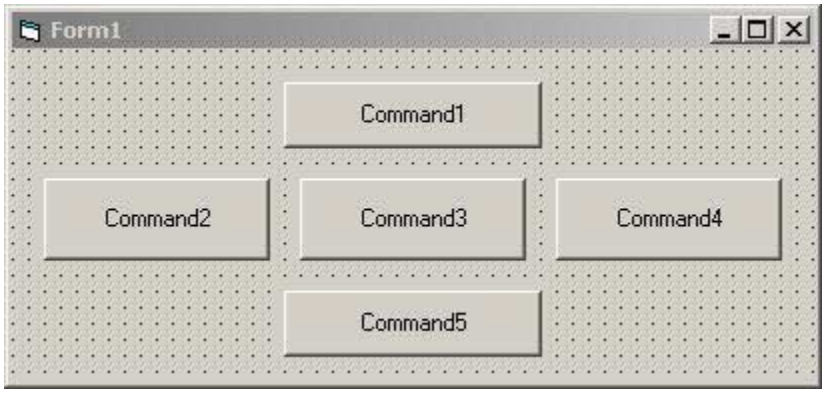
5 – Terminer Test_VBMI

Cliquer sur ce bouton ferme tout ce qui a trait à cet exemple.

B - Préparation d’un EXE avec VisualBasic

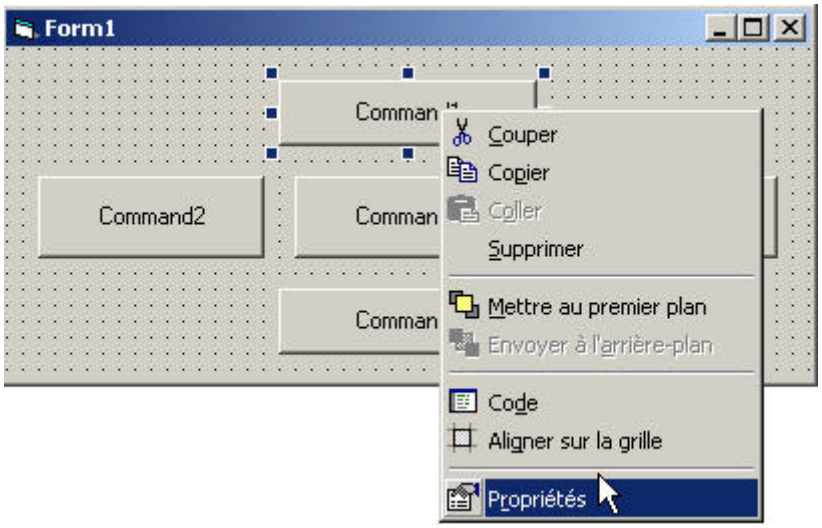
0 – Ouvrir un nouveau projet VB

1 – Préparer une feuille (form) avec 5 boutons de commande



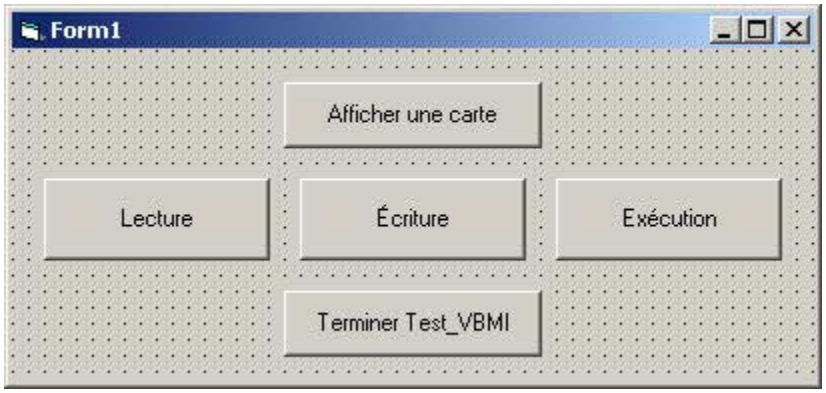
2 – Donnez une définition pour chaque bouton

a/ clic « droit » sur le bouton puis « Propriétés »



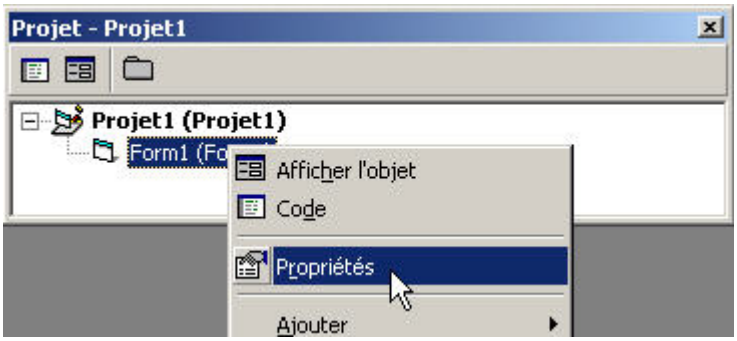
b/ Inscrivez dans « caption » le titre du bouton

c/ Complétez les autres boutons pour obtenir

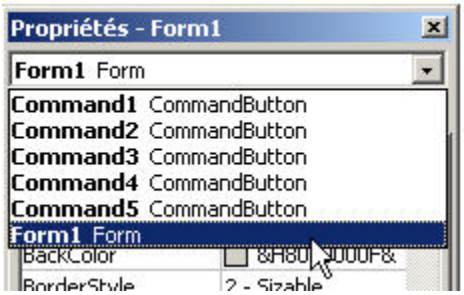


3 – Donnez un nom à cette feuille. Ce nom sera celui par lequel cette application VB sera reconnue.

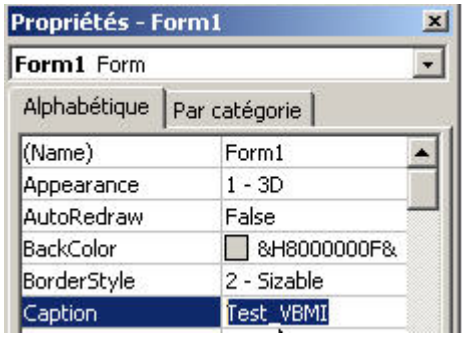
a/ clic « droit » sur « Form1 » puis « Propriétés » si la fenêtre Propriétés n'est pas ouverte



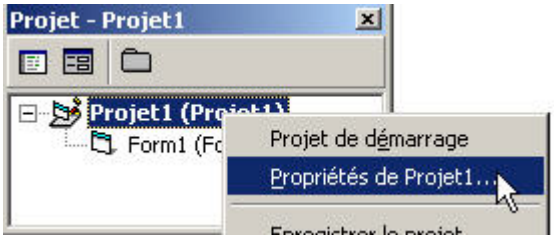
b/ sélectionnez Form1 dans la liste déroulante



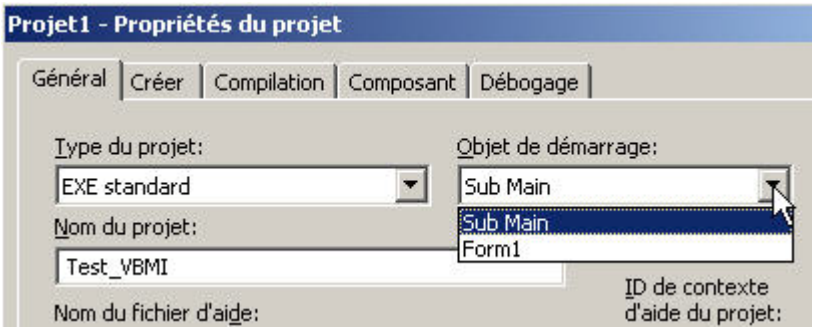
c/ Entrez le titre sous « caption »



4 – Clic « droit » sur Projet1 ouvre les « propriétés » du projet

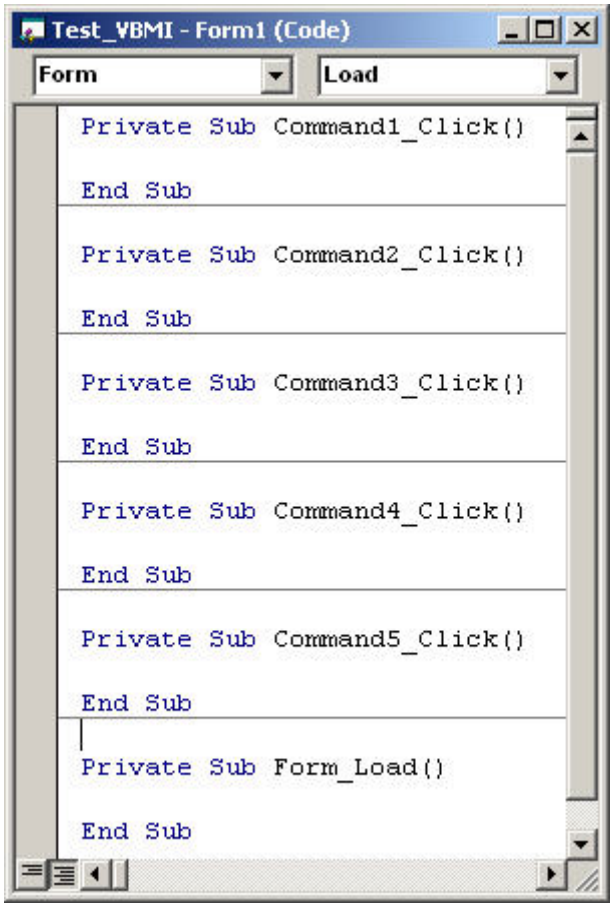


Entrez le nom et choisissez comme objet de démarrage « Sub Main »



5 – Ajoutez le code « donnant des choses à faire » quand les boutons sont activés.

Doubles clics sur la feuille (en dehors des boutons) puis sur chaque bouton donnent cette fenêtre :

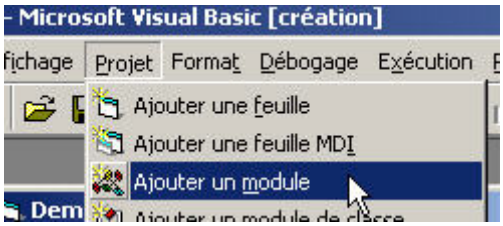


Si Form_Load() existe, supprimez cette sub. Il faut ensuite ajouter du code pour obtenir le texte suivant.

NOTER que la référence complète et le nom de la carte à afficher peuvent être modifiés.

```
Private Sub Command1_Click()
   carto.Do "open table ""c:\mes documents\polysens.tab"" map from polysens"
End Sub

Private Sub Command2_Click()
   MsgBox VarList("Var1").Value & " / " & VarList("Var2").Value
End Sub

Private Sub Command3_Click()
   carto.MBApplications(NumApp).Do ("Bonjour")
   VarList("Var1").Value = 0
   VarList("Var2").Value = "Bibi"
End Sub

Private Sub Command4_Click()
   VarList("Var1").Value = 2
   MsgBox carto.MBApplications(NumApp).Eval("Function1")
End Sub

Private Sub Command5_Click()
   carto.Do "close all"
   carto.Do "Terminate Application ""Test_VBMI.Mbx"" "
   Unload Me
End Sub
```

6 – Ajoutez un module (menu Projet | Ajouter un module)



ce qui ouvre une fenêtre où entrer du code

Code à entrer ::

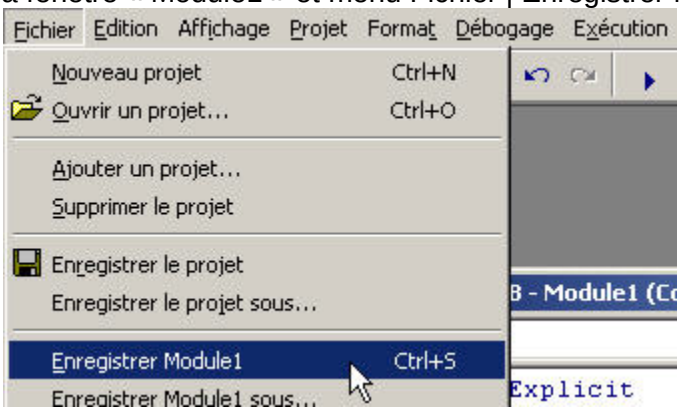
```
Option Explicit

Private Declare Function SetWindowWord Lib "user32" (ByVal hwnd As Long, ByVal
nIndex As Long, ByVal wNewWord As Long) As Long

Global AppList As Object
Global VarList As Object
Global hNwMap As Long
```

```
Global carto As Object
Global NumApp As Integer

Sub Main()

Set carto = GetObject(, "MapInfo.Application")
LoadMbx
OuvrirWin
End Sub

Public Sub OuvrirWin()
If hNwMap = 0 Then
Exit Sub
End If
Form1.Show
Call SetWindowWord(Form1.hwnd, -8, hNwMap)
End Sub

Public Sub LoadMbx()

Dim i As Integer
Set AppList = carto.MBApplications

If AppList.Count > 0 Then
   For i = 1 To AppList.Count
      If AppList(i).Name = "Test_VBMI.MBX" Then
        NumApp = i
        Set VarList = carto.MBApplications(NumApp).MBGlobals
        hNwMap = VarList("hNwMap")
       Exit Sub
      End If
   Next
End If
End Sub
```

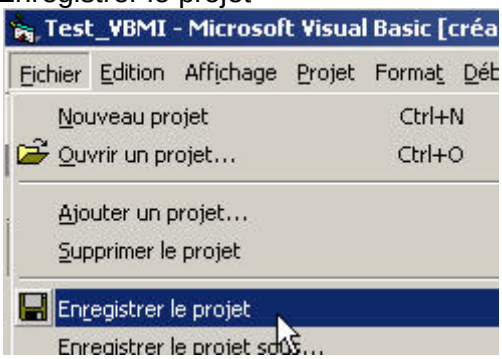
7 – Sauvegarde des composantes

a/ Activez la fenêtre « principale » et menu Fichier | Enregistrer Form1

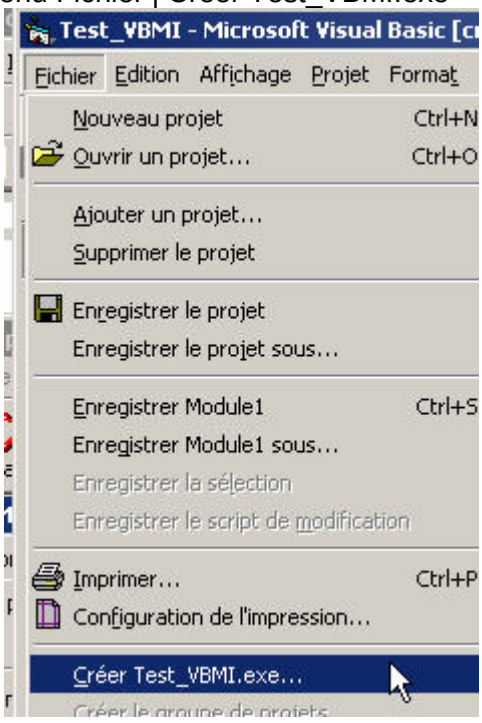
b/ Activez la fenêtre « Module1 » et menu Fichier | Enregistrer Module1



c/ menu Fichier | Enregistrer le projet



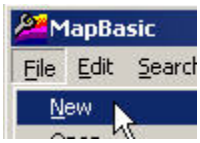
8 – Compilez le projet : menu Fichier | Créer Test_VBMI.exe



C - Préparation d'un MBX avec MapBasic

MapMasic

1 – Ouvrez une nouvelle page



2 – Entrez le code suivant

```
include "MapBasic.def"

Declare sub main

Declare sub RemoteMsgHandler
Declare function RemoteQueryHandler() as string
Declare function carre(ByVal N as float) as Float

Declare sub au_sujet_de
Declare sub fin

Global hNwMap as integer

Global Var1 as Integer
Global Var2 as String

Sub main

    Create menu "Test" as
          "Au sujet de Test_VBMI" calling au_sujet_de,
             "(-",
             "Terminer Text_VBMI.mbx" calling fin

    Alter menu bar add "Test"
    var1=999
    var2="Toto"
    Run Program  ApplicationDirectory$() + "Test_VBMI.exe"
    hNwMap=WindowInfo(WIN_MAPINFO   ,WIN_INFO_WND)
end sub

sub RemoteMsgHandler()
```

```
        Note commandinfo(1000)
    end sub

    function RemoteQueryHandler() as string
        if commandinfo(1000)="Function1" then
            RemoteQueryHandler="Le carré de " & str$(var1) & " est " & Carre(Var1)
        end if
    end function

    function carre(ByVal N as float) as Float
        Carre=N * N
    end function

    sub au_sujet_de
        Note "Test_VBMI démontre les principes de base de l'intégration d'une fenêtre VB6"+
             " dans une application MapBasic."+chr$(10)+chr$(10)+
             "Basé sur des documents préparés par Laurent Maumet et Sébastien Roddier"
    end sub

    sub fin
        note "L'application Test_VBMI.MBX va être fermée."+chr$(10+
            "Assurez-vous que la fenêtre VB l'est aussi avant de continuer."
        terminate application "Test_VBMI.MBX"
    end sub
```

3 – Sauvegardez en donnant le nom Test_VBMI.MB (important car ce nom sera utilisé par le programme VB pour reconnaître l'application MBX)

4 – Compilez

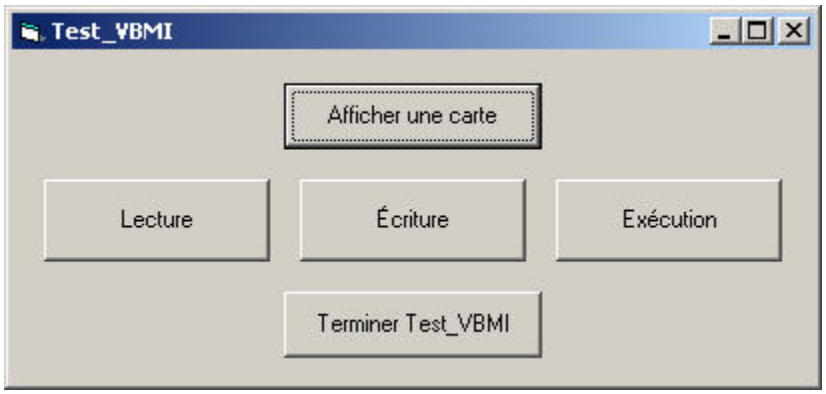
D - Utilisation dans MapInfo

0 – Les fichiers Test_VBMI.MBX, Test_VBMI.EXE doivent se trouver dans le même répertoire. Les fichiers de la table à afficher (POLYSENS dans cet exemple) doivent se trouver dans le répertoire tel que spécifié dans « Sub Command1_Click() »

1 – Lancez Test_VBMI.MBX

2 – L'application installe un petit menu dans la barre principale

et la fenêtre VB



3 – Pour terminer, utiliser le bouton « Terminer Test_VBMI » de préférence; les deux applications MBX et EXE seront fermées ainsi que la carte qui aurait été ouverte.

Si vous utilisez le menu « Terminer Test_VBMI.MBX, seul le MBX sera fermé

E – Explications

Certaines explications sont aussi données dans les portions de code listées pdans ce document.

Code associé aux boutons de Form1

Cmd 1

```
carto.Do "open table ""c:\mes documents\polysens.tab"" map from polysens"
```

Façon typique de passer une commande à MapInfo depuis VB. Carto est défini dans Module1 comme étant l'objet MapInfo

Cdm 2

```
MsgBox VarList("Var1").Value & " / " & VarList("Var2").Value
```

Message VB qui liste les valeurs de Var1 et Var2 définies comme global dans le code MB

Cmd 3

```
carto.MBApplications(NumApp).Do ("Bonjour")
```

Message affiché par MapInfo

```
VarList("Var1").Value = 0
VarList("Var2").Value = "Bibi"
```

Mise à jour des valeurs des variables définies dans MapBasic

Cmd 4

```
VarList("Var1").Value = 2
MsgBox carto.MBApplications(NumApp).Eval("Function1")
```

Mise à jour de la valeur de la variable Var2 et affichage dans VB du résultat de l'appel à une fonction définie dans notre application MB.

Cmd 5

```
carto.Do "close all"
carto.Do "Terminate Application ""Test_VBMI.Mbx"" "
```

Commandes MapInfo pour fermer la fenêtre et l'application MBX

```
Unload Me
```

Command VB pour terminer l'EXE..

Code associé au module Module1

```
Option Explicit

Private Declare Function SetWindowWord Lib "user32" (ByVal hwnd As Long, ByVal nIndex As Long, ByVal wNewWord As Long) As Long
```

```
Global carto As Object
```
Déclaration de l'objet MapInfo

```
Global hNwMap As Long
```
Référence Windows (Handle) de la fenêtre MapInfo

```
Global AppList As Object
```
Déclaration de l'objet pour la collection de MBX en mémoire

```
Global NumApp As Integer
```
Numéro de l'application Test_VBMI.MBX

```
Global VarList As Object
```
Déclaration de l'objet pour la collection de variables globales en mémoire

Sub Main()

```
Set carto = GetObject(, "MapInfo.Application")
```
Chargement de l'objet MapInfo
```
LoadMbx
```
Appel à une sub pour construire la liste des applications MBX en mémoire
```
OuvrirWin
```
Appel à une sub pour rendre la fenêtre fille de MapInfo
```
End Sub
```

Public Sub OuvrirWin()

```
If hNwMap = 0 Then
  Exit Sub
End If
Form1.Show
```
Affiche la fenêtre VB
```
Call SetWindowWord(Form1.hwnd, -8, hNwMap)
```
Api pour rendre une fenêtre fille d une application, ici Mapinfo
```
End Sub
```

Public Sub LoadMbx()

```
Dim i As Integer
Set AppList = carto.MBApplications
```
Construit une liste des mbx qui tourne en mémoire

```
If AppList.Count > 0 Then
 For i = 1 To AppList.Count
   If AppList(i).Name = "Test_VBMI.MBX" Then '
```
Tester pour trouver le nom de l'application mapbasic qui appelle ce programme
```
     NumApp = i
     Set VarList = carto.MBApplications(NumApp).MBGlobals
```
Lecture de la variable pour le Handle
```
     hNwMap = VarList("hNwMap")
```

```
      Exit Sub
    End If
  Next
End If
End Sub
```

Code MapBasic

```
include "MapBasic.def"

Declare sub main

Declare sub RemoteMsgHandler
Declare function RemoteQueryHandler() as string
```

Modules requis pour saisir l'appel à une fonction MB

```
Declare function carre(ByVal N as float) as Float
```

Fonction qui sera "appelée"

```
Declare sub au_sujet_de
Declare sub fin
```

Sub "cosmétiques"

```
Global hNwMap as integer
```

Le Handle de l'application

```
Global Var1 as Integer
Global Var2 as String
```

Deux variables globales dont le contenu seru lu et modifié

Sub main

```
    Create menu "Test" as
          "Au sujet de Test_VBMI" calling au_sujet_de,
             "(-",
             "Terminer Text_VBMI.mbx" calling fin

    Alter menu bar add "Test"
    var1=999
    var2="Toto"

    Run Program  ApplicationDirectory$() + "Test_VBMI.exe"
```

Lancement du programme

```
    hNwMap=WindowInfo(WIN_MAPINFO   ,WIN_INFO_WND)
```

Récupération du handle de MapInfo en global

```
end sub
```

sub RemoteMsgHandler()

```
     Note commandinfo(1000)
```

Saisie du « nom » de la fonction dans le message envoyé par VB

```
end sub


function RemoteQueryHandler() as string
    if commandinfo(1000)="Function1" then
            Reconnaissance du “nom” de la fonction demandée et action en
            conséquence
        RemoteQueryHandler="Le carré de " & str$(var1) & " est " & Carre(Var1)
    end if
end function

function carre(ByVal N as float) as Float
    Carre=N * N
end function

sub au_sujet_de
    Note "Test_VBMI démontre les principes de base de l'intégration d'une fenêtre VB6"+
         " dans une application MapBasic."+chr$(10)+chr$(10)+
         "Basé sur des documents préparés par Laurent Maumet et Sébastien Roddier"
end sub

sub fin
    note "L'application Test_VBMI.MBX va être fermée."+chr$(10+
        "Assurez-vous que la fenêtre VB l‘est aussi avant de continuer."
    terminate application "Test_VBMI.MBX"
end sub
```