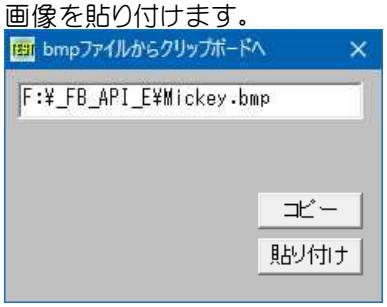
クリップボードへ転送と取り出し(BITMAP) II

BITMAPファイルを読み込み、クリップボードへ転送します。また、そのデータをピクチャボックスに描画します。
LoadImage 画像ファイルの読み込み
CopyImage イメージを拡大縮小してコピー
OpenClipboard クリップボードをオープン
EmptyClipboard クリップボードをからにする
SetClipboardData クリップボードにデータを設定
GetClipboardData クリップボードから指定フォーマットのデータを検索
CloseClipboard クリップボードをクローズ
IsClipboardFormatAvailable 指定したフォーマットがクリップボードにあるかどうかを判定
CreateCompatibleDC メモリデバイスコンテキストを作成
SelectObject 指定されたデバイスコンテキストのオブジェクトを選択
GetObject オブジェクトを取得
BitBlt ビットブロック転送を行う
DeleteDC 指定されたデバイスコンテキストを削除
GetDC デバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放

BitmapファイルをEditBoxにドラッグ&ドロップし「コピー」クリックでクリップボードに転送、「貼り付け」でフォームに画像を貼り付けます。




```
'=======================================
'= クリップボードへ転送と取り出し(BITMAP) II
'=    (Clipboard.bas)
'=======================================
#include "Windows.bi"

Type BITMAP
    bmType          As Long
    bmWidth         As Long
    bmHeight        As Long
    bmWidthBytes    As Long
    bmPlanes        As Integer
    bmBitsPixel     As Integer
    bmBits          As Long
End Type

#define LR_LOADFROMFILE &H10            '外部ファイルからロードする
#define IMAGE_BITMAP 0                  'ビットマップ
#define IMAGE_CURSOR 2                  'カーソル
#define IMAGE_ENHMETAFILE 3             '拡張メタファイル
#define IMAGE_ICON 1                    'アイコン
#define CF_TEXT 1                       'テキスト形式のデータ。各行は復帰改行(CR-LF)コードで
                                        終わる
#define CF_BITMAP 2                     'ビットマップのデータ(HBITMAP)
#define SRCCOPY &HCC0020                'そのまま転送
```

```
' 画像ファイルの読み込み
Declare Function Api_LoadImage& Lib "user32" Alias "LoadImageA" (ByVal hInst&, ByVal
lpszName$, ByVal uType&, ByVal cxDesired&, ByVal cyDesired&, ByVal fuLoad&)

' クリップボードをオープン
Declare Function Api_OpenClipboard& Lib "User32" Alias "OpenClipboard" (ByVal hWnd&)

' クリップボードをクローズ
Declare Function Api_CloseClipboard& Lib "User32" Alias "CloseClipboard" ()

' クリップボードを空にする
Declare Function Api_EmptyClipboard& Lib "user32" Alias "EmptyClipboard" ()

' クリップボードにデータを設定
Declare Function Api_SetClipboardData& Lib "user32" Alias "SetClipboardData" (ByVal
wFormat&, ByVal hMem&)

' クリップボードから指定フォーマットのデータを検索
Declare Function Api_GetClipboardData& Lib "user32" Alias "GetClipboardData" (ByVal
wFormat&)

' 指定したフォーマットがクリップボードにあるかどうか判定
Declare Function Api_IsClipboardFormatAvailable& Lib "user32" Alias
"IsClipboardFormatAvailable" (ByVal wFormat&)

' ビットブロック転送を行う。コピー元からコピー先のデバイスコンテキストへ、指定された長方形内の各ピクセルの色
データをコピー
Declare Function Api_BitBlt& Lib "gdi32" Alias "BitBlt" (ByVal hDestDC&, ByVal X&, ByVal
Y&, ByVal nWidth&, ByVal nHeight&, ByVal hSrcDC&, ByVal xSrc&, ByVal ySrc&, ByVal dwRop&)

' 指定されたデバイスコンテキストに関連するデバイスと互換性のあるメモリデバイスコンテキストを作成
Declare Function Api_CreateCompatibleDC& Lib "gdi32" Alias "CreateCompatibleDC" (ByVal
hDC&)

' オブジェクト取得
Declare Function Api_GetObject& Lib "gdi32" Alias "GetObjectA" (ByVal hObject&, ByVal
nCount&, lpObject As Any)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

' 指定されたデバイスコンテキストのオブジェクトを選択
Declare Function Api_SelectObject& Lib "gdi32" Alias "SelectObject" (ByVal hDC&, ByVal
hObject&)

' 指定されたデバイスコンテキストを削除
Declare Function Api_DeleteDC& Lib "gdi32" Alias "DeleteDC" (ByVal hDC&)

Var Shared Edit1 As Object
Var Shared Button1 As Object
Var Shared Button2 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14

Var Shared hDC As Long
Var Shared mhDC As Long
Var Shared FileName As String

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()

    hDC = Api_GetDC(GethWnd)
```

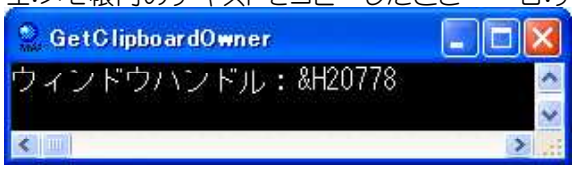
```
End sub

'=====================================
'= シェルドロップされたファイル名を取得
'=====================================
Declare Sub Edit1_DropFiles edecl (ByVal DF As Long)
Sub Edit1_DropFiles(ByVal DF As Long)
    Var CN As Long

    CN = GetDropFileCount(DF)
    FileName = GetDropFileName(DF, 0)
    Edit1.SetWindowText FileName
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hBitmap As Long
    Var Ret As Long

    'ビットマップを指定するサイズでメモリにロード
    hBitmap = Api_LoadImage(GethInst, FileName, IMAGE_BITMAP, 100, 100, LR_LOADFROMFILE)
    If hBitmap = 0 Then
        A% = MessageBox(GetWindowText, "ビットマップをロードできません！", 0, 2)
        Exit Sub
    End If

    'クリップボードをオープン
    Ret = Api_OpenClipboard(GethWnd)

    'クリップボードをクリア
    Ret = Api_EmptyClipboard()

    'クリップボードにビットマップをセット
    Ret = Api_SetClipboardData(CF_BITMAP, hBitmap)

    'クリップボードをクローズ
    Ret = Api_CloseClipboard()
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var bmp As BITMAP
    Var hBit As Long
    Var Ret As Long

    'Bitmap型式データの有無を調査
    If Api_IsClipboardFormatAvailable(CF_BITMAP) <> 0 Then
        Ret = Api_OpenClipboard(GethWnd)

        '指定フォーマットのBITMAPデータを検索
        hBit = Api_GetClipboardData(CF_BITMAP)

        'メモリデバイスコンテキストを作成
        mhDC = Api_CreateCompatibleDC(hDC)

        'Object取得
        Ret = Api_GetObject(hBit, Len(bmp), bmp)

        'Object選択
        Ret = Api_SelectObject(mhDC, hBit)

        '指定ののデバイスコンテキストにメモリデバイスコンテキストのデータを転送
        Ret = Api_BitBlt(hDC, 10, 40, bmp.bmWidth, bmp.bmHeight, mhDC, 0, 0, SRCCOPY)
    End If
```

```
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
    Var Ret As Long

    Ret = Api_ReleaseDC(GethWnd, hDC)
    Ret = Api_DeleteDC(mhDC)
    Ret = Api_CloseClipboard()
End SUb

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

クリップボードを所有するウィンドウのハンドルを取得

GetClipboardOwner クリップボードを所有するウィンドウのハンドルを取得

左:メモ帳内のテキストをコピーしたとき　　右:デスクトップ画面を「PrintScreen」でコピーしたとき



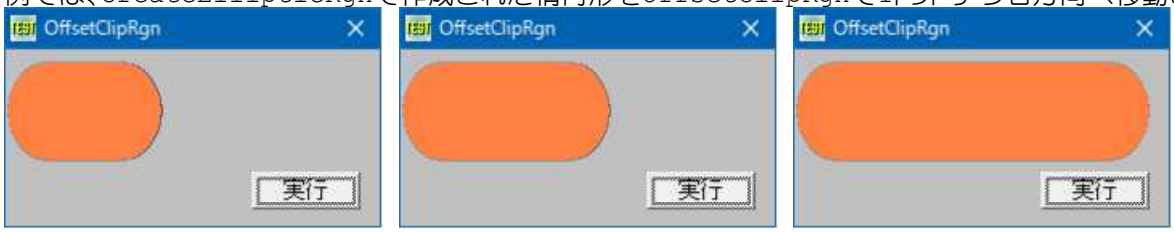

```
'=====================================
'= クリップボードを所有しているウィンドウのハンドルを取得
'=    (GetClipboardOwner.bas)
'=====================================

' クリップボードを所有するウィンドウのハンドルを取得
Declare Function Api_GetClipboardOwner& Lib "user32" Alias "GetClipboardOwner" ()

Var Owner As Long

Owner = Api_GetClipboardOwner()

If Owner <> 0 Then
    Print "ウィンドウハンドル:&H" & Hex$(Owner)
Else
    Print "所有権を持つウィンドウはありません！"
End If

Stop
End
```

クリップ領域を指定されたオフセット分移動

OffsetClipRgn デバイス コンテキストのクリップ領域を指定されたオフセット分移動
DeleteObject オブジェクトに関連付けられていたすべてのシステムリソースを解放
SelectObject 指定されたデバイスコンテキストのオブジェクトを選択
Rectangle 長方形の描画
CreateSolidBrush ソリッドカラーで論理ブラシを作成
SelectClipRgnクリッピング領域を設定

CreateEllipticRgn 楕円形のリージョンを作成
GetClientRect ウィンドウのクライアント領域の座標を取得
Sleep カレントスレッドの実行を指定の時間だけ中断
GetDC デバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放

例では、CreateEllipticRgnで作成された楕円形をOffsetClipRgnで1ドットずつ右方向へ移動させています。




```
'=======================================
'= クリップ領域を指定されたオフセット分移動
'=    (OffsetClipRgn.bas)
'=======================================
#include "Windows.bi"

Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

' ウィンドウのクライアント領域の座標を取得
Declare Function Api_GetClientRect& Lib "user32" Alias "GetClientRect" (ByVal hWnd&,
lpRect As RECT)

' 楕円形のリージョンを作成
Declare Function Api_CreateEllipticRgn& Lib "gdi32" Alias "CreateEllipticRgn" (ByVal
X1&, ByVal Y1&, ByVal X2&, ByVal Y2&)

' クリッピング領域を設定
Declare Function Api_SelectClipRgn& Lib "gdi32" Alias "SelectClipRgn" (ByVal hDC&, ByVal
hRgn&)

' ソリッドカラーで論理ブラシを作成
Declare Function Api_CreateSolidBrush& Lib "gdi32" Alias "CreateSolidBrush" (ByVal
crColor&)

' 指定されたデバイスコンテキストのオブジェクトを選択
Declare Function Api_SelectObject& Lib "gdi32" Alias "SelectObject" (ByVal hDC&, ByVal
hObject&)

' デバイス コンテキストのクリップ領域を指定されたオフセット分移動
Declare Function Api_OffsetClipRgn& Lib "gdi32" Alias "OffsetClipRgn" (ByVal hDC&, ByVal
x&, ByVal y&)

' ペン、ブラシ、フォント、ビットマップ、リージョン、パレットのいずれかの論理オブジェクトを削除し、そのオブジェクトに
関連付けられていたすべてのシステムリソースを解放。オブジェクトを削除した後は、指定されたハンドルは無効になる
Declare Function Api_DeleteObject& Lib "gdi32" Alias "DeleteObject" (ByVal hObject&)

' 長方形の描画
Declare Function Api_Rectangle& Lib "gdi32" Alias "Rectangle" (ByVal hDC&, ByVal X1&,
ByVal Y1&, ByVal X2&, ByVal Y2&)

' カレントスレッドの実行を指定の時間だけ中断
Declare Sub Api_Sleep Lib "Kernel32" Alias "Sleep" (ByVal dwMilliseconds&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)
```

```
Var Shared Button1 As Object

Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hDC As Long
    Var rc As RECT
    Var Rgn As Long
    Var Brush As Long
    Var Prev As Long
    Var Cnt As Long
    Var Ret As Long

    Randomize Time

    Cls

    'デバイスコンテキスト取得
    hDC = Api_GetDC(GethWnd)

    'ウィンドウのクライアント領域の座標を取得
    Ret = Api_GetClientRect(GethWnd, rc)

    '楕円形のリージョンを作成
    Rgn = Api_CreateEllipticRgn(0, 8, 50, 70)

    'クリッピング領域を設定
    Ret = Api_SelectClipRgn(hDC, Rgn)

    'ソリッドカラーで論理ブラシを作成(ランダム)
    Brush = Api_CreateSolidBrush(RGB(Rnd(1) * 255, Rnd(1) * 255, Rnd(1) * 255))

    '指定されたデバイスコンテキストのオブジェクトを選択
    Prev = Api_SelectObject(hDC, Brush)

    '座標(0 ～ Right-50)の範囲
    For Cnt = 0 To rc.Right - 50 Step 1

        'デバイス コンテキストのクリップ領域を指定されたオフセット分移動
        Ret = Api_OffsetClipRgn(hDC, 1, 0)

        '長方形の描画
        Ret = Api_Rectangle(hDC, rc.Left, rc.Top, rc.Right, rc.Bottom)

        '時間調整
        Api_Sleep 8
    Next Cnt

    'オブジェクト・デバイスコンテキストの解放
    Ret = Api_SelectObject(hDC, Prev)
    Ret = Api_DeleteObject(Brush)
    Ret = Api_DeleteObject(Rgn)
    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

グレースケール･セピア変換

フルカラー画像をYIQ変換しグレースケール･セピアで表示します。
CreateCompatibleDC 互換性のあるメモリデバイスコンテキストを作成
CreateCompatibleBitmap デバイスコンテキストと互換性のあるビットマップを作成
SelectObject 指定されたデバイスコンテキストのオブジェクトを選択
DeleteDC 指定されたデバイスコンテキストを削除
DeleteObject グラフィックスオブジェクトを削除し、システムリソースを解放
BitBlt 画像のビットブロック転送
GetPixel ピクセルのカラー値を取得
SetPixelV 指定の位置のピクセルを指定のカラーに最も近いカラー値に設定

YIQ変換
YIQ変換とは、カラー画像の表色系の一つで、**Y** は輝度、**I** は肌色を含むオレンジからシアンにかけての色調、**Q** はそれ以外の色を表す。
RGBからYIQ変換を行うには次の式で表される。
Y=0.299*R+0.587*G+0.114*B

```
'=====================================
'= グレースケールセピア変換
'=    (GrayScale.bas)
'=====================================
#include "Windows.bi"

' 指定されたデバイスコンテキストに関連するデバイスと互換性のあるメモリデバイスコンテキストを作成
Declare Function Api_CreateCompatibleDC& Lib "gdi32" Alias "CreateCompatibleDC" (ByVal
hDC&)

' デバイスコンテキストと互換性のあるビットマップを作成
Declare Function Api_CreateCompatibleBitmap& Lib "gdi32" Alias "CreateCompatibleBitmap"
(ByVal hDC&, ByVal nWidth&, ByVal nHeight&)

' 指定されたデバイスコンテキストのオブジェクトを選択
Declare Function Api_SelectObject& Lib "gdi32" Alias "SelectObject" (ByVal hDC&, ByVal
hObject&)

' 指定されたデバイスコンテキストを削除
Declare Function Api_DeleteDC& Lib "gdi32" Alias "DeleteDC" (ByVal hDC&)

' ペン、ブラシ、フォント、ビットマップ、リージョン、パレットのいずれかの論理オブジェクトを削除し、そのオブジェクトに
関連付けられていたすべてのシステムリソースを解放します。オブジェクトを削除した後は、指定されたハンドルは無効
になります。
Declare Function Api_DeleteObject& Lib "gdi32" Alias "DeleteObject" (ByVal hObject&)

' ビットブロック転送を行います。コピー元からコピー先のデバイスコンテキストへ、指定された長方形内の各ピクセル
の色データをコピーします。
Declare Function Api_BitBlt& Lib "gdi32" Alias "BitBlt" (ByVal hDestDC&, ByVal X&, ByVal
Y&, ByVal nWidth&, ByVal nHeight&, ByVal hSrcDC&, ByVal xSrc&, ByVal ySrc&, ByVal dwRop&)

' 指定された座標のピクセルのRGB値を取得
Declare Function Api_GetPixel& Lib "gdi32" Alias "GetPixel" (ByVal hDC&, ByVal X&, ByVal
Y&)

' 指定の位置のピクセルを指定のカラーに最も近いカラー値に設定
Declare Function Api_SetPixelV& Lib "gdi32" Alias "SetPixelV" (ByVal hDC&, ByVal X&,
ByVal Y&, ByVal crColor&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得。
その後、GDI 関数を使って、返されたデバイスコンテキスト内で描画を行える
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)
```

```
' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

#define SRCCOPY &HCC0020                    'そのまま転送
#define DSTINVERT &H550009                  'コピー先を反転してコピー
Var Shared Picture1 As Object
Var Shared Picture2 As Object
Var Shared Radio1 As Object
Var Shared Radio2 As Object
Var Shared Button1 As Object

Picture1.Attach GetDlgItem("Picture1")
Picture2.Attach GetDlgItem("Picture2")
Radio1.Attach GetDlgItem("Radio1") : Radio1.SetFontSize 14
Radio2.Attach GetDlgItem("Radio2") : Radio2.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var Shared Bitmap As Object
BitmapObject Bitmap

Var Shared hDC1 As Long
Var Shared hDC2 As Long
Var Shared Flag As Byte

'=====================================
'=
'=====================================
Declare Function GraySepiaBltSub(ByVal lngColor As Long) As Long
Function GraySepiaBltSub(ByVal lngColor As Long) As Long
    On Error Goto *Er_Trap

    Var Ret As Long

    '色をRGBに分割し、YIQ変換を実行
    Ret = (lngColor And &HFF) * 0.299 + ((lngColor And &HFF00) / 256) * 0.587 + ((lngColor
And &HFF0000) / 65536) * 0.114

    'RGB値が0～255の範囲に収まるようにする
    If Ret >= 255 Then
        Ret = 255
    Else If Ret <= 0 Then
        Ret = 0
    End If

    '変換した値を返す
    If Flag = 0 Then
        GraySepiaBltSub = RGB(Ret, Ret, Ret)              'グレースケール
    Else
        GraySepiaBltSub = RGB(Ret + 30, Ret, Ret - 30)    'セピア
    End If
    Exit Function

*Er_Trap
    Resume Next
End Function

'=====================================
'=
'=====================================
Declare Sub GraySepiaBlt(ByVal srchDC As Long, ByVal srcLeft As Long, ByVal srcTop As
Long, ByVal srcWidth As Long, ByVal srcHeight As Long, ByVal hDC2 As Long, ByVal trgLeft As
Long, ByVal trgTop As Long )
Sub GraySepiaBlt(ByVal srchDC As Long, ByVal srcLeft As Long, ByVal srcTop As Long, ByVal
srcWidth As Long, ByVal srcHeight As Long, ByVal hDC2 As Long, ByVal trgLeft As Long, ByVal
trgTop As Long )
    On Error Goto *Er_Trap

    Var lX As Long
    Var lY As Long
    Var Ret As Long
```

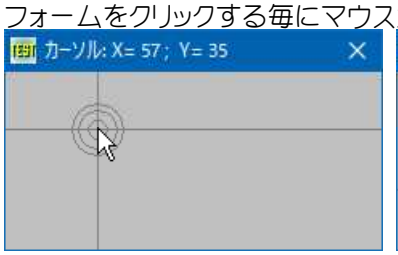
```
    Var mhDC As Long
    Var hBmp As Long

    'メモリDCを作成
    mhDC = Api_CreateCompatibleDC(srchDC)

    'ビットマップオブジェクトの作成
    hBmp = Api_CreateCompatibleBitmap(srchDC, srcWidth, srcHeight)

    'メモリDCへビットマップを割付
    Ret = Api_SelectObject(mhDC, hBmp)

    'メモリDCへ画像をコピー
    Ret = Api_BitBlt(mhDC, 0, 0, srcWidth, srcHeight, srchDC, srcLeft, srcTop, SRCCOPY)

    '色情報の読み込み及びYIQ変換処理
    For lY = 0 To srcHeight - 1
        For lX = 0 To srcWidth - 1
            Ret = Api_SetPixelV(mhDC, lX, lY, GraySepiaBltSub(Api_GetPixel(mhDC, lX,
lY)))
        Next lX
    Next lY

    '対象のDCにメモリDCをコピー
    Ret = Api_BitBlt(hDC2, trgLeft, trgTop, srcWidth, srcHeight, mhDC, 0, 0, SRCCOPY)

*Er_Trap

    'メモリDCの削除
    Ret = Api_DeleteDC(mhDC)

    'ビットマップオブジェクトの削除
    Ret = Api_DeleteObject(hBmp)
End Sub

'=====================================
'=
'=====================================
Declare Sub Mainform_Start edecl ()
Sub mainform_Start()

    'Picture1にBitmap読込
    Bitmap.LoadFile "flower.bmp"
    Picture1.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()

    'Picture1のデバイスコンテキスト取得
    hDC1 = Api_GetDC(Picture1.GethWnd)
    hDC2 = Api_GetDC(Picture2.GethWnd)

    If Radio1.GetCheck = 1 Then Flag = 0 Else Flag = 1

    SetMousePointer 2
    Picture2.Cls

    'グレースケール･セピア変換実行
    GraySepiaBlt hDC1, 0, 0, Picture1.GetWidth, Picture1.GetHeight, hDC2, 0, 0

    SetMousePointer 0

    'デバイスコンテキスト解放
    Ret = Api_ReleaseDC(Picture1.GethWnd, hDC1)
    Ret = Api_ReleaseDC(Picture2.GethWnd, hDC2)
```

```
End Sub

'====================================
'=
'====================================
While 1
    WaitEvent
Wend
Stop
End
```

クロスヘアカーソル

クロスヘアカーソルを描画します。
ShowCursor マウスカーソルの表示・非表示

フォームをクリックする毎にマウスカーソルを表示・非表示します。



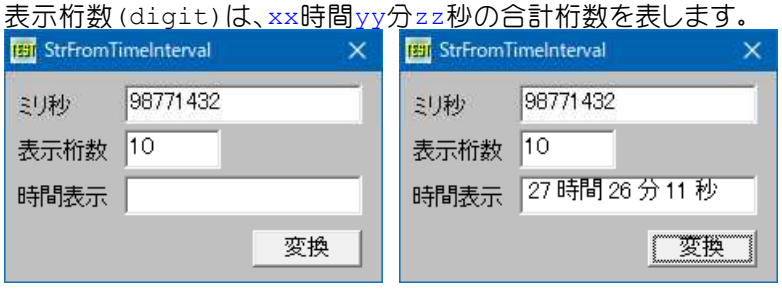

```
'====================================
'= クロスヘアカーソル
'=    (CrossHairCursor.bas)
'====================================
#include "Windows.bi"

' マウスカーソルの表示・非表示
Declare Function Api_ShowCursor& Lib "user32" Alias "ShowCursor" (ByVal bShow&)

Var Shared MainForm As Object

MainForm.Attach GethWnd

'====================================
'=
'====================================
Declare Sub HideMouse ()
Sub HideMouse()
    While Api_ShowCursor(0) >= 0
    Wend
End Sub

'====================================
'=
'====================================
Declare Sub ShowMouse ()
Sub ShowMouse()
    While Api_ShowCursor(1) < 0
    Wend
End Sub

'====================================
'=
'====================================
Declare Sub MainForm_MouseMove edecl (ByVal Button As Integer, ByVal Shift As Integer,
ByVal x As Single, ByVal y As Single)
Sub MainForm_MouseMove(ByVal Button As Integer, ByVal Shift As Integer, ByVal x As Single,
ByVal y As Single)
    Cls

    Line (x, 0) - (x, GetHeight), , 1
```

```
    Line (0, y) - (GetWidth, y), , 1
    Circle (x, y), 6, 1
    Circle (x, y), 12, 1
    Circle (x, y), 16, 1
    SetWindowText "カーソル: X=" & Str$(x) & " ;  Y=" & Str$(y)
End Sub

'=======================================
'=
'=======================================
Declare Sub MainForm_Click edecl ()
Sub MainForm_Click()
    static CursorAn As Integer
    CursorAn = Not CursorAn

    If CursorAn Then
        HideMouse
    Else
        ShowMouse
    End If
End Sub

'=======================================
'=
'=======================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
    ShowMouse
    End
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

経過時間(ミリ秒)をフォーマットして表示

ミリ秒で表された数値を経過時間フォーマットで表示します。
StrFromTimerInterval 時間を表す数値を文字列に変換

表示桁数(digit)は、xx時間yy分zz秒の合計桁数を表します。

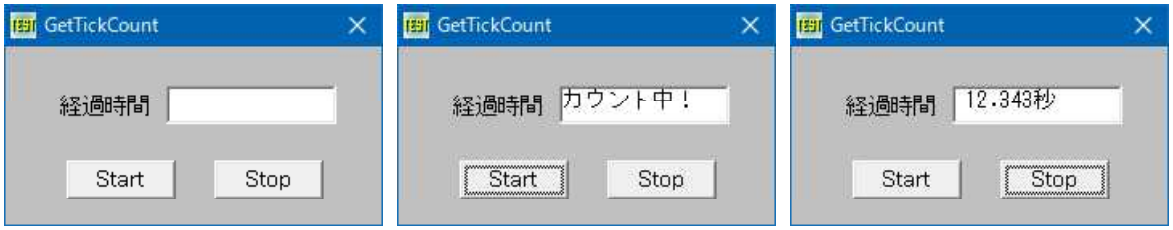



```
'=======================================
'= 経過時間(ミリ秒)をフォーマットして表示
'=    (StrFromTimeInterval2.bas)
'=======================================
#include "Windows.bi"

' 時間を表す数値を文字列に変換
Declare Function Api_StrFromTimeInterval& Lib "shlwapi" Alias "StrFromTimeIntervalA"
(ByVal pszOut$, ByVal cchMax&, ByVal dwTimeMS&, ByVal dwDigits&)

Var Shared Text(3) As Object
```

```
Var Shared Edit(1) As Object
Var Shared Button1 As Object

For i = 0 To 3
    If i < 2 Then
        Edit(i).Attach GetDlgItem("Edit" & Trim$(Str$(i + 1))) : Edit(i).SetFontSize 14
    End If
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1))) : Text(i).SetFontSize 14
Next
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

ShowWindow -1
Cls

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var wTime As Long
    Var digit As Integer
    Var sOut As String
    Var Ret As Long

    Text(3).SetWindowText ""

    sOut = String$(100, 0)

    'ミリ秒取得
    wTime = Val(Edit(0).GetWindowText)

    'xx時間yy分zz秒 表示するxxyyzzの桁数取得
    digit = Val(Edit(1).GetWindowText)

    Ret = Api_StrFromTimeInterval(sOut, Len(sOut), wTime, digit)
    Text(3).SetWindowText sOut
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

経過時間を取得

GetTickCount システムが起動してからの経過時間を取得



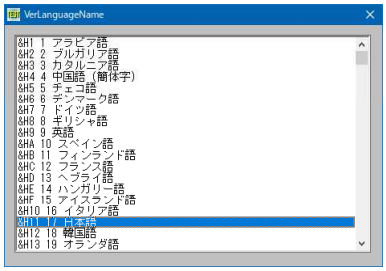
参考
GetTickCount関数は、Windowsが起動されてから約49.7日間継続すると「0」に戻る。
また、Long型で扱うため約25日経過すると値がマイナス値になるので、値の大小を比較する必要があります。

```
'=====================================
'= 経過時間を取得
'=    (GetTickCount.bas)
'=====================================
```

```
#include "Windows.bi"

' システムが起動してからの経過時間を取得
Declare Function Api_GetTickCount& Lib "Kernel32" Alias "GetTickCount" ()

Var Shared Text(1) As Object
Var Shared Button(1) As Object

For i = 0 To 1
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1))) : Text(i).SetFontSize 14
    Button(i).Attach GetDlgItem("Button" & Trim$(Str$(i + 1))) : Button(i).SetFontSize 14
Next i

Var Shared Count As Long

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Text(1).SetWindowText "カウント中！"

    Count = Api_GetTickCount()
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Count2 As Long

    Count2 = Api_GetTickCount()

    '通常
    If Count2 > Count Then
        Text(1).SetWindowText Str$((Count2 - Count) / 1000) & "秒"

    '25日間を超えた場合
    Else
        Text(1).SetWindowText Str$((Count - Count2) / 1000) & "秒"
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

言語IDから言語名を取得

VerLanguageName 言語IDから言語名を取得



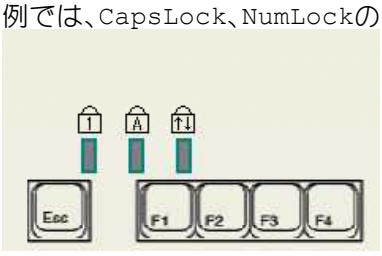
該当のないIDでは Buffer に「ニュートラル言語」が入るのでそれ以外の言語を ListBox に表示させています。

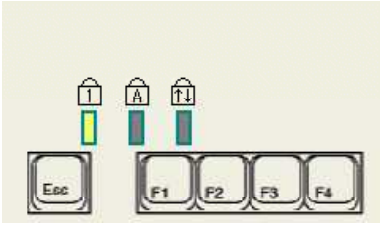
```
'=====================================
'= 言語IDから言語名を取得
'=    (VerLanguageName.bas)
'=====================================
#include "Windows.bi"

' 言語IDから言語名を取得
Declare Function Api_VerLanguageName& Lib "kernel32" Alias "VerLanguageNameA" (ByVal
wLang&, ByVal szLang$, ByVal nSize&)

Var Shared List1 As Object
List1.Attach GetDlgItem("List1") : List1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var Buffer As String
    Var LangID As Long
    Var Ret As Long

    For LangID = 0 To 20490
        Buffer = String$(255, 0)
        Ret = Api_VerLanguageName(LangID, Buffer, Len(Buffer))

        Buffer =  Left$(Buffer, InStr(1, Buffer, Chr$(0)) - 1)
        If Buffer <> "ニュートラル言語" Then
            List1.AddString "&H" & hex$(LangID) & " " & Trim$(Str$(LangID)) & " " & Buffer
        End If
    Next
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

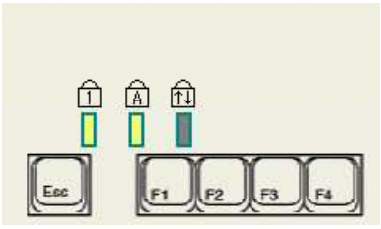
現在のキーの状態を調べる

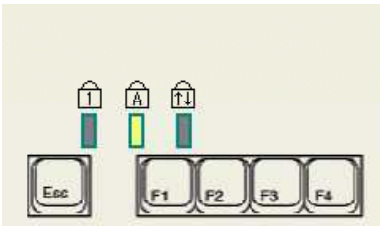
GetKeyState 現在のキーの状態を調べる

例では、CapsLock、NumLockのON/OFF状態を取得しています。







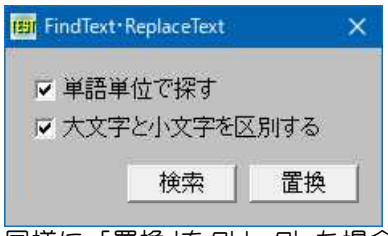


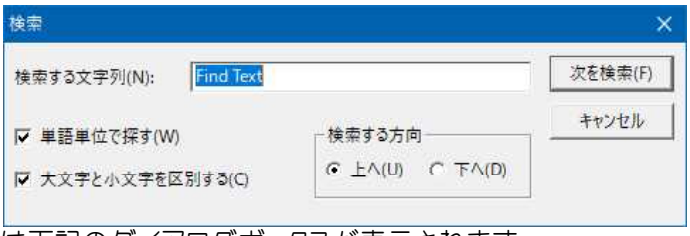

```
'=====================================
'= 現在のキーの状態を調べる
'=    (GetKeyState2.bas)
'=====================================
#include "Windows.bi"

' 現在のキーの状態を調べる
Declare Function Api_GetKeyState& Lib "user32" Alias "GetKeyState" (ByVal nVirtKey&)

#define vbKeyCapital 20                    'CapsLockキー
#define vbKeyNumlock 144                   'NumLockキー

Var Shared Text(3) As Object

For i = 0 To 3
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1)))
    Text(i).SetFontSize 14
Next

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    If Api_GetKeyState(vbKeyCapital) = 1 Then
        Text(2).SetWindowText "ON"
    Else
        Text(2).SetWindowText "OFF"
    End If

    If Api_GetKeyState(vbKeyNumlock) = 1 Then
        Text(3).SetWindowText "ON"
    Else
        Text(3).SetWindowText "OFF"
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

検索・置換ダイアログボックスの作成

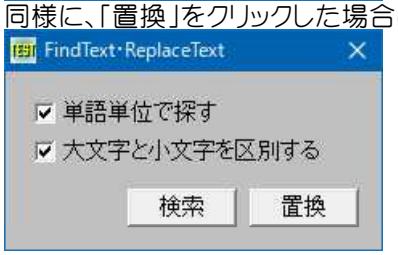
FindText システムが定義したモードレスの［検索］ダイアログボックスを作成

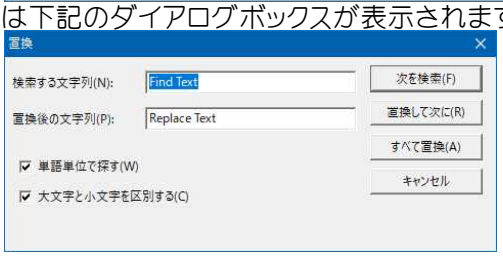
ReplaceText システムが定義したモードレスの［置換］ダイアログボックスを作成





同様に、「置換」をクリックした場合は下記のダイアログボックスが表示されます。





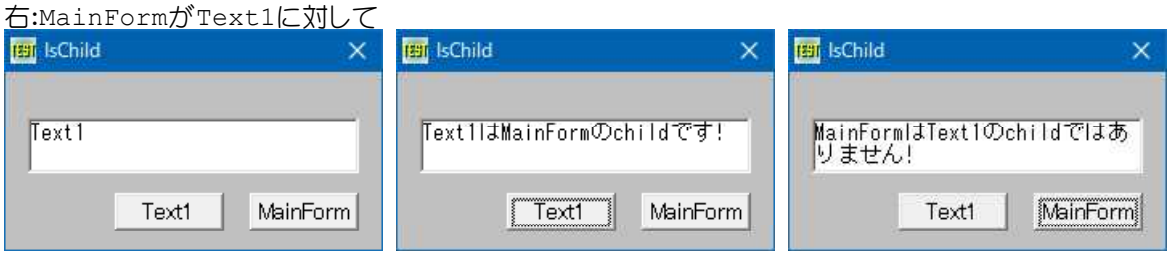

```
'=======================================
'=「検索」･「置換」ダイアログボックスの作成
'=    (FindText.bas)
'=======================================
#include "Windows.bi"

Type FINDREPLACE
    lStructSize       As Long
    hwndOwner         As Long
    hInstance         As Long
    flags             As Long
    lpstrFindWhat     As Long
    lpstrReplaceWith  As Long
    wFindWhatLen      As Integer
    wReplaceWithLen   As Integer
    lCustData         As Long
    lpfnHook          As Long
    lpTemplateName    As Long
End Type

' システムが定義したモードレスの［検索］ダイアログボックスを作成
Declare Function Api_FindText& Lib "comdlg32" Alias "FindTextA" (pFindreplace As
FINDREPLACE)

' システムが定義したモードレスの［置換］ダイアログボックスを作成
Declare Function Api_ReplaceText& Lib "comdlg32" Alias "ReplaceTextA" (pFindreplace As
FINDREPLACE)

Var Shared Check(1) As Object
Var Shared Button(1) As Object

For i = 0 To 1
    Check(i).Attach GetDlgItem("Check" & Trim$(Str$(i + 1))) : Check(i).SetFontSize 14
    Button(i).Attach GetDlgItem("Button" & Trim$(Str$(i + 1))) : Button(i).SetFontSize 14
Next

Var Shared fr As FINDREPLACE
Var Shared c1 As Long
Var Shared c2 As Long

'=======================================
'=
'=======================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    fr.lpstrReplaceWith = StrAdr("Replace Text" & Chr$(0))
    fr.lpstrFindWhat = StrAdr("Find Text" & Chr$(0))
    fr.wFindWhatLen = 9
    fr.wReplaceWithLen = 12
    fr.hInstance = GethInst
```

```
    fr.hwndOwner = GethWnd
    fr.lStructSize = Len(fr)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long
    If Check(0).GetCheck = 1 Then c1 = 2 Else c1 = 0
    If Check(1).GetCheck = 1 Then c2 = 4 Else c2 = 0
    fr.flags = c1 Or c2

    Ret = Api_FindText(fr)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Ret As Long

    If Check(0).GetCheck = 1 Then c1 = 2 Else c1 = 0
    If Check(1).GetCheck = 1 Then c2 = 4 Else c2 = 0
    fr.flags = c1 Or c2

    Ret = Api_ReplaceText(fr)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

子ウィンドウかどうかを調べる

指定のウィンドウに対して、子ウィンドウかどうかを調べます。
IsChild ウィンドウが子ウィンドウかどうかを判断

左:MainformにText1を貼り付けています
中:Text1がMainFormに対して
右:MainFormがText1に対して

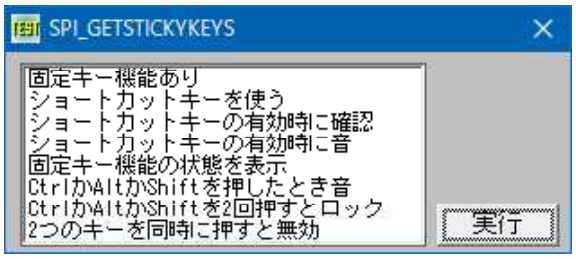



```
'=====================================
'= 子ウィンドウかどうかを調べる
'=    (IsChild.bas)
'=====================================
#include "Windows.bi"

' ウィンドウが子ウィンドウかどうかを判断
Declare Function Api_IsChild& Lib "user32" Alias "IsChild" (ByVal hWndParent&, ByVal
hWnd&)
```

```
Var Shared Text1 As Object
Var Shared Button1 As Object
Var Shared Button2 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    If Api_IsChild(GethWnd, Text1.GethWnd) = 1 Then
        Text1.SetWindowText "Text1はMainFormのchildです!"
    Else
        Text1.SetWindowText "Text1はMainFormのchildではありません!"
    End If
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    If Api_IsChild(Text1.GethWnd, GethWnd) = 1 Then
        Text1.SetWindowText "MainFormはText1のchildです!"
    Else
        Text1.SetWindowText "MainFormはText1のchildではありません!"
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

固定キー機能の情報を取得

SystemParametersInfo システム全体に関するパラメータを取得・設定




```
'=====================================
'= 固定キー機能の情報を取得
'=    (SPI_GETSTICKYKEYS.bas)
'=====================================
#include "Windows.bi"

Type tagSTICKYKEYS
    cbSize      As Long
    dwFlags     As Long
End Type

#define SKF_AUDIBLEFEEDBACK &H40            'このフラグが設定されると、システムは、ユーザーが、ロッ
```

```
                                                ク、ラッチ、サウンド、またはリリース修飾キーは
#define SKF_AVAILABLE &H2                       '固定キー機能が使用可能
#define SKF_CONFIRMHOTKEY &H8                   'Windows9*・Windows2000:確認のダイアログボックスが
                                                固定キー機能のホットキーを使用してアクティブに表示・
#define SKF_HOTKEYACTIVE &H4                    'ユーザーは、固定キー、Shiftキーを5回押してオンとオフ
                                                の機能を無効にすることができる
#define SKF_HOTKEYSOUND &H10                    'システムは、ユーザーが固定ホットキーを使用してオンまた
                                                はオフ機能をオンサイレンの音を再生する
#define SKF_INDICATOR &H20                      'Win98・Win2K:視覚的なインジケータは、固定キー機能が
                                                表示される必要があります。
#define SKF_LALTLATCHED &H10000000              'Win98・Win2K:左[Alt]キーがラッチされる
#define SKF_LALTLOCKED &H100000                 'Win98・Win2K:左[Alt]キーがロックされる
#define SKF_LCTLLATCHED &H4000000               'Win98・Win2K:左[Ctrl]キーがラッチされる
#define SKF_LCTLLOCKED &H40000                  'Win98・Win2K:左[Ctrl]キーがロックされる
#define SKF_LSHIFTLATCHED &H1000000             'Win98・Win2K:左[Shift]キーがラッチされる
#define SKF_LSHIFTLOCKED &H10000                'Win98・Win2K:左[Shift]キーがロックされる
#define SKF_LWINLATCHED &H40000000              'Win98・Win2K:左[Windows]キーがラッチされる
#define SKF_LWINLOCKED &H400000                 'Win98・Win2K:左[Windows]キーがロックされる
#define SKF_RALTLATCHED &H20000000              'Win98・Win2K:右[Alt]キーがラッチされる
#define SKF_RALTLOCKED &H200000                 'Win98・Win2K:右[Alt]キーがロックされる
#define SKF_RCTLLATCHED &H8000000               'Win98・Win2K:右[Ctrl]キーがラッチされる
#define SKF_RCTLLOCKED &H80000                  'Win98・Win2K:右[Ctrl]キーがロックされる
#define SKF_RSHIFTLATCHED &H2000000             'Win98・Win2K:右[Shift]キーがラッチされる
#define SKF_RSHIFTLOCKED &H20000                'Win98・Win2K:右[Shift]キーがロックされる
#define SKF_RWINLATCHED &H80000000              'Win98・Win2K:右[Windows]キーがラッチされる
#define SKF_RWINLOCKED &H800000                 'Win98・Win2K:右[Windows]キーがロックされる
#define SKF_STICKYKEYSON &H1                    '固定キー機能が有効
#define SKF_TRISTATE &H80                       '[Shift]・[Ctrl]・[Alt]を2回押したとき、押し続けてい
                                                ると見なし3回目を押したとき、その設定を解除
#define SKF_TWOKEYSOFF &H100                    '別のキーと同時に押している[Shift]・[Ctrl]・[Alt]を
                                                アップしたとき、スティックキーをオフにする
#define SPI_GETSTICKYKEYS 58                    'ユーザー補助機能の固定キーを定義するSTICKYKEYS構
                                                造体取得
' システム全体に関するパラメータを取得・設定
Declare Function Api_SystemParametersInfo& Lib "user32" Alias "SystemParametersInfoA"
(ByVal uiAction&, ByVal uiParam&, pvParam As Any, ByVal fWinIni&)

Var Shared List1 As Object
Var Shared Button1 As Object

Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
List1.Attach GetDlgItem("List1") : List1.SetFontSize 12

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var tsk As tagSTICKYKEYS
    Var Ret As Long

    'リストボックスをクリア
    List1.ResetContent

    '構造体を初期化
    tsk.cbSize = Len(tsk)

    '固定キー機能の情報を取得
    Ret = Api_SystemParametersInfo(SPI_GETSTICKYKEYS, Len(tsk), tsk, 0)

    '固定キー機能の情報を表示
    If (tsk.dwFlags And SKF_STICKYKEYSON) = SKF_STICKYKEYSON Then
        List1.AddString "固定キー機能有効"
    End If

    If (tsk.dwFlags And SKF_AVAILABLE) = SKF_AVAILABLE Then
        List1.AddString "固定キー機能あり"
    End If

    If (tsk.dwFlags And SKF_HOTKEYACTIVE) = SKF_HOTKEYACTIVE Then
```

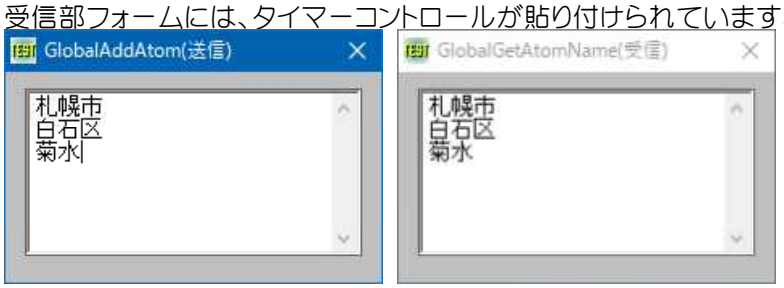
```
        List1.AddString "ショートカットキーを使う"
    End If

    If (tsk.dwFlags And SKF_CONFIRMHOTKEY) = SKF_CONFIRMHOTKEY Then
        List1.AddString "ショートカットキーの有効時に確認"
    End If

    If (tsk.dwFlags And SKF_HOTKEYSOUND) = SKF_HOTKEYSOUND Then
        List1.AddString "ショートカットキーの有効時に音"
    End If

    If (tsk.dwFlags And SKF_INDICATOR) = SKF_INDICATOR Then
        List1.AddString "固定キー機能の状態を表示"
    End If

    If (tsk.dwFlags And SKF_AUDIBLEFEEDBACK) = SKF_AUDIBLEFEEDBACK Then
        List1.AddString "CtrlかAltかShiftを押したとき音"
    End If

    If (tsk.dwFlags And SKF_TRISTATE) = SKF_TRISTATE Then
        List1.AddString "CtrlかAltかShiftを2回押すとロック"
    End If

    If (tsk.dwFlags And SKF_TWOKEYSOFF) = SKF_TWOKEYSOFF Then
        List1.AddString "2つのキーを同時に押すと無効"
    End If

    If (tsk.dwFlags And SKF_LALTLOCKED) = SKF_LALTLOCKED Then
        List1.AddString "左Altキー ロック"
    End If

    If (tsk.dwFlags And SKF_LCTLLOCKED) = SKF_LCTLLOCKED Then
        List1.AddString "左Ctrlキー ロック"
    End If

    If (tsk.dwFlags And SKF_LSHIFTLOCKED) = SKF_LSHIFTLOCKED Then
        List1.AddString "左Shiftキー ロック"
    End If

    If (tsk.dwFlags And SKF_RALTLOCKED) = SKF_RALTLOCKED Then
        List1.AddString "右Altキー ロック"
    End If

    If (tsk.dwFlags And SKF_RCTLLOCKED) = SKF_RCTLLOCKED Then
        List1.AddString "右Ctrlキー ロック"
    End If

    If (tsk.dwFlags And SKF_RSHIFTLOCKED) = SKF_RSHIFTLOCKED Then
        List1.AddString "右Shiftキー ロック"
    End If

    If (tsk.dwFlags And SKF_LWINLOCKED) = SKF_LWINLOCKED Then
        List1.AddString "左Windowsキー ロック"
    End If

    If (tsk.dwFlags And SKF_RWINLOCKED) = SKF_RWINLOCKED Then
        List1.AddString "右Windowsキー ロック"
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

異なるアプリケーション間の通信

異なるアプリケーション間の通信をテストします。
送信部
SetProp ウィンドウに関するプロパティを設定
RemoveProp 指定のウィンドウプロパティリストから項目を削除
GlobalAddAtom グローバルアトムテーブルに項目を追加
GlobalDeleteAtom グローバルアトムテーブルから指定のアトムの項目を削除

受信部
GetProp ウィンドウに関連するプロパティを取得する
FindWindowEx クラス名 or キャプションを与えてウィンドウのハンドルを取得
GlobalGetAtomName グローバルアトムテーブルから文字列を取得

送信部エディットボックスに入力したテキストは、受信部で周期的にチェックされ受信側エディットボックスに表示されます。
受信部フォームには、タイマーコントロールが貼り付けられています。




```
'=======================================
'= GlobalAddAtom(テキスト送信)
'=    (GlobalAddAtom.bas)
'=======================================
#include "Windows.bi"

' ウィンドウに関するプロパティを設定
Declare Function Api_SetProp& Lib "user32" Alias "SetPropA" (ByVal hWnd&, ByVal
lpString$, ByVal hData&)

' 指定のウィンドウプロパティリストから項目を削除
Declare Function Api_RemoveProp& Lib "user32" Alias "RemovePropA" (ByVal hWnd&, ByVal
lpString$)

' グローバルアトムテーブルに項目を追加
Declare Function Api_GlobalAddAtom% Lib "kernel32" Alias "GlobalAddAtomA" (ByVal
lpString$)

' グローバルアトムテーブルから指定のアトムの項目を削除
Declare Function Api_GlobalDeleteAtom% Lib "kernel32" Alias "GlobalDeleteAtom" (ByVal
nAtom%)

Var Shared intAtom As Integer

Var Shared Edit1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14

'=======================================
'=
'=======================================
Declare Sub Edit1_Change edecl ()
Sub Edit1_Change()
    Var strText As String
    Var lngRet As Long
    Var intRet As Integer

    If intAtom <> 0 Then
        intRet = Api_GlobalDeleteAtom(intAtom)
        intAtom = 0
    End If
```

```
    strText = Left$(GetDlgItemText("Edit1"), 255)
    intAtom = Api_GlobalAddAtom(strText)
    If intAtom <> 0 Then
        lngRet = Api_SetProp(GetHwnd, "ExportProp", intAtom)
    End If
End Sub

'=====================================
'=
'=====================================
Declare Sub Mainform_QueryClose edecl (Cancel%, Mode%)
Sub Mainform_QueryClose(Cancel%, Mode%)
    Var lngRet As Long
    Var intRet As Integer

    If Cancel% = 0 Then
        If intAtom <> 0 Then
            intRet = Api_GlobalDeleteAtom(intAtom)
        End If
        lngRet = Api_RemoveProp(GetHwnd, "ExportProp")
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

```
'=====================================
'= GlobalGetAtomName(テキスト受信)
'=    (GlobalGetAtomName.bas)
'=====================================
#include "Windows.bi"

' ウィンドウに関連するプロパティを取得
Declare Function Api_GetProp& Lib "user32" Alias "GetPropA" (ByVal hWnd&, ByVal
lpString$)

' クラス名 、または キャプションを与えてウィンドウのハンドルを取得
Declare Function Api_FindWindowEx& Lib "user32" Alias "FindWindowExA" (ByVal
hWndParent&, ByVal hWndChildAfter&, ByVal lpszClass$, ByVal lpszWindow$)

' グローバルアトムテーブルから文字列を取得
Declare Function Api_GlobalGetAtomName% Lib "kernel32" Alias "GlobalGetAtomNameA" (ByVal
nAtom%, ByVal lpBuffer$, ByVal nSize&)

Var Shared Edit1 As Object
Var Shared Timer1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Timer1.Attach GetDlgItem("Timer1")

'=====================================
'=
'=====================================
Declare Sub Mainform_Start edecl ()
Sub Mainform_Start()

    Timer1.SetInterval 30
    Timer1.Enable -1
End Sub
```

```
'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var intRet As Integer
    Var intAtom As Integer
    Var hWnd As Long
    Var strText As String * 255                 '文字列の長さは255バイト以下

    hWnd = Api_FindWindowEx(0, 0, ByVal 0, "GlobalAddAtom(送信)")

    If hWnd Then
        intAtom = Api_GetProp(hWnd, "ExportProp")
        If intAtom Then
            intRet = Api_GlobalGetAtomName(intAtom, strText, 255)
            Edit1.SetWindowText strText
        End If
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

コピー・切り取り・クリア

SendMessage ウィンドウにメッセージを送信
WM_COPY(&H301) テキストボックス・コンボボックスの選択テキストをクリップボードにコピー
WM_CUT(&H300) 選択されているテキスト部分を削除、そのテキストをCF_TEXTフォーマットでクリップボードにコピー
WM_CLEAR(&H303) テキストボックス・コンボボックスの選択テキストを削除
CF_TEXT(1) テキスト形式のデータ。各行は復帰改行(CR-LF)コードで終わる


```
'=====================================
'= コピー・切り取り・クリア
'=    (SendMessage8.bas)
'=====================================
#include "Windows.bi"

' ウィンドウにメッセージを送信
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

' クリップボードをオープン
Declare Function Api_OpenClipboard& Lib "user32" Alias "OpenClipboard" (ByVal hWnd&)

' クリップボードから指定フォーマットのデータを検索
Declare Function Api_GetClipboardData& Lib "user32" Alias "GetClipboardData" (ByVal
wFormat&)
```

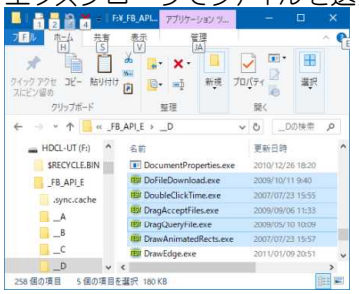
```
' クリップボードをクローズ
Declare Function Api_CloseClipboard& Lib "user32" Alias "CloseClipboard" ()

' ヒープに確保されたメモリをロック
Declare Function Api_GlobalLock& Lib "kernel32" Alias "GlobalLock" (ByVal hMem&)

' メモリブロックのロックを解除
Declare Function Api_GlobalUnlock& Lib "kernel32" Alias "GlobalUnlock" (ByVal hMem&)

' 文字列をコピーする
Declare Function Api_lstrcpy& Lib "kernel32" Alias "lstrcpy" (ByVal lpString1 As Any,
ByVal lpString2 As Any)

#define WM_COPY &H301                    'テキストボックス・コンボボックスの選択テキストをクリップボードにコピー
#define WM_CUT &H300                     '選択されているテキスト部分を削除し、そのテキストをCF_TEXTフォーマットでクリップボードにコピー
#define WM_CLEAR &H303                   'テキストボックス・コンボボックスの選択テキストを削除
#define CF_TEXT 1                        'テキスト形式のデータ。各行は復帰改行(CR-LF)コードで終わる

#define MAXSIZE 4096

Var Shared Edit1 As Object
Var Shared Text1 As Object
Var Shared Radio(2) As Object
Var Shared Button1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
For i = 0 To 2
    Radio(i).Attach GetDlgItem("Radio" & Trim$(Str$(i + 1)))
    Radio(i).SetFontSize 14
Next

'=====================================
'=
'=====================================
Declare Function Index bdecl () As Integer
Function Index()
    Index = Val(Mid$(GetDlgRadioSelect("Radio1"), 6)) - 1
End Function

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Edit1.SetWindowText "F-Basic Programming Tips"

    '"Programming "を選択
    Edit1.SetSelText 8, 20
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hMem As Long
    Var lpMem As Long
    Var txt As String
    Var Ret As Long

    ClearCB

    Select Case Index
        Case 0                           'コピー
            Ret = Api_SendMessage(Edit1.GethWnd, WM_COPY, 0, ByVal 0)
        Case 1                           '切り取り
```

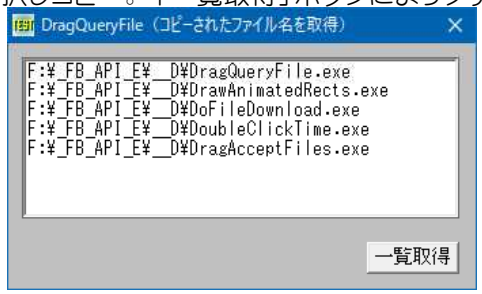
```
            Ret = Api_SendMessage(Edit1.GethWnd, WM_CUT, 0, ByVal 0)
        Case 2                                  'クリア
            Ret = Api_SendMessage(Edit1.GethWnd, WM_CLEAR, 0, ByVal 0)
    End Select

    Text1.SetWindowText ""
    If Api_OpenClipboard(0) = 0 Then
        Text1.SetWindowText "クリップボードが開きません"
        Exit Sub
    End If

    'テキストを参照しているグローバルメモリのブロックへのハンドルを取得
    hMem = Api_GetClipboardData(CF_TEXT)

    'クリップボードのメモリをロックし、実際の文字列を参照
    lpMem = Api_GlobalLock(hMem)

    If lpMem <> 0 Then
        txt = Space$(MAXSIZE)
        Ret = Api_lstrcpy(txt, lpMem)
        Ret = Api_GlobalUnlock(hMem)

        'nullを削除
        txt = Left$(txt, InStr(txt, Chr$(0)) - 1)
        Text1.SetWindowText txt
    Else
        Text1.SetWindowText "文字列を参照できません！"
    End If

    Ret = Api_CloseClipboard()
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

コピーされたファイル名をクリップボードから取得

マイコンピュータやエクスプローラ等でコピーされたファイル名をクリップボードから取得します。
IsClipboardFormatAvalable 指定したフォーマットがクリップボードにあるかどうかの判定
OpenClipboard クリップボードをオープン
GetClipboardData クリップボードから指定のフォーマットのデータを検索
CloseClipboard クリップボードをクローズ
DragQueryFile ドラッグアンドドロップされたファイル名を取得
lstlen 指定された文字列のバイトまたは文字の長さを返す

エクスプローラでファイルを選択しコピー。「一覧取得」ボタンによりクリップボードからのファイルを取得





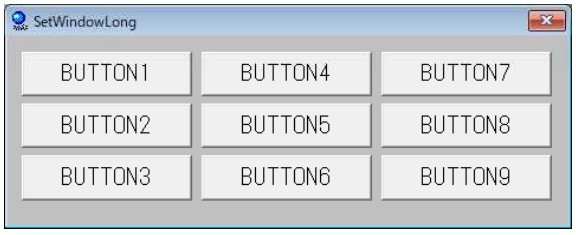

```
'=======================================
'= コピーされたファイル名をクリップボードから取得
'=    (DragQueryFile.bas)
'=======================================
```

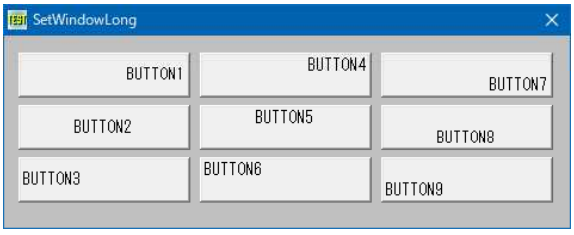
```
#include "Windows.bi"

' 指定したフォーマットがクリップボードにあるかどうか判定
Declare Function Api_IsClipboardFormatAvailable& Lib "user32" Alias
"IsClipboardFormatAvailable" (ByVal wFormat&)

' クリップボードをオープン
Declare Function Api_OpenClipboard& Lib "user32" Alias "OpenClipboard" (ByVal hWnd&)

' クリップボードから指定フォーマットのデータを検索
Declare Function Api_GetClipboardData& Lib "user32" Alias "GetClipboardData" (ByVal
wFormat&)

' クリップボードをクローズ
Declare Function Api_CloseClipboard& Lib "user32" Alias "CloseClipboard" ()

'ドラッグアンドドロップされたファイル名を取得
Declare Function Api_DragQueryFile& Lib "shell32" Alias "DragQueryFileA" (ByVal hDrop&,
ByVal iFile&, ByVal lpszFile$, ByVal cch&)

' 指定された文字列のバイトまたは文字の長さを返す
Declare Function Api_lstrlen& Lib "Kernel32" Alias "lstrlenA" (ByVal lpString$)

#define CF_HDROP 15                          'HDROP型

Var Shared List1 As Object
Var Shared Button1 As Object
List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'= クリップボードにCF_HDROP形式データがあるかどうかの確認
'=====================================
Declare Function ClipboardHasFiles() As Integer
Function ClipboardHasFiles() As Integer

    ClipboardHasFiles = (Api_IsClipboardFormatAvailable(CF_HDROP) > 0)
End Function

'=====================================
'= Chr$(0)を取り除く
'=====================================
Declare Function TrimNull(startstr As String) As String
Function TrimNull(startstr As String) As String
   TrimNull = Left$(startstr, Api_lstrLen(startstr))
End Function

'=====================================
'= クリップボードのCF_HDROP形式ファイルデータをListBoxに
'=====================================
Declare Sub ClipboardGetFiles()
Sub ClipboardGetFiles()
    Var hCBData As Long                      'クリップボード内のデータハンドル
    Var numFiles As Long                     'データの数
    Var cbBuffer As Long
    Var buffer As String
    Var cnt As Long
    Var Ret As Long

    List1.ResetContent

    'CF_HDROP形式ファイルデータがある場合
    If ClipboardHasFiles() Then

        'クリップボードオープン
        If Api_OpenClipboard(0) <> 0 Then

            'クリップボードデータのハンドルを取得
            hCBData = Api_GetClipboardData(CF_HDROP)
```

```
            '指定形式のデータが存在する場合
            If hCBData <> 0 Then

                'ドラッグドロップされたデータ数を取得
                numFiles = Api_DragQueryFile(hCBData, -1, ByVal 0, 0)

                For cnt = 0 To numFiles - 1
                    cbBuffer = Api_DragQueryFile(hCBData, cnt, ByVal 0, 0)

                    If cbBuffer > 0 Then
                        buffer = Space$(cbBuffer + 1)
                        cbBuffer = Len(buffer)

                        'Bufferに入ったファイル名を整形(TrimNull)してListBoxに表示
                        If Api_DragQueryFile(hCBData, cnt, buffer, cbBuffer) > 0 Then
                            List1.AddString TrimNull(buffer)
                        End If
                    End If
                Next cnt
            End If

            'クリップボードクローズ
            Ret = Api_CloseClipboard
        End If
    End If
End Sub

'=======================================
'= エクスプローラ等でコピーされたファイル名をクリップボードから取得
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()

    ClipboardGetFiles
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

コマンドボタン内の文字位置設定

コマンドボタン内の文字位置を設定します。
GetWindowLong 指定されたウィンドウの属性を取得
SetWindowLong 指定されたウィンドウの属性を変更






```
'=======================================
'= コマンドボタン内の文字位置設定
'=    (SetWindowLong.bas)
'=======================================
#include "Windows.bi"

#define BS_LEFT &H100                    'ボタンの中にテキストを左揃え
```

```
#define BS_RIGHT &H200                          'ボタンの中にテキストを右揃え
#define BS_CENTER &H300                         'ボタンの中にテキストを中央揃え
#define BS_TOP &H400                            'ボタンの上部にテキストを置く
#define BS_BOTTOM &H800                         'ボタンの下部にテキストを置く
#define GWL_STYLE -16                           'アプリケーションのインスタンスハンドル

' 指定されたウィンドウの属性を変更
Declare Function Api_SetWindowLong& Lib "user32" Alias "SetWindowLongA" (ByVal hWnd&,
ByVal nIndex&, ByVal dwNewLong&)

' 指定されたウィンドウの属性を取得
Declare Function Api_GetWindowLong& Lib "user32" Alias "GetWindowLongA" (ByVal hWnd&,
ByVal nIndex&)

Var Shared Button(8) As Object

For i = 0 To 8
    Button(i).Attach GetDlgItem("Button" & Trim$(Str$(i + 1)))
    Button(i).SetFontSize 14
Next

'=====================================
'=
'=====================================
Declare Sub Mainform_Start edecl ()
Sub Mainform_Start()
    Var Ret As Long

    For i = 0 To 8
        Ret = Api_GetWindowLong(Button(i).GethWnd, GWL_STYLE)
        Select Case i
            Case 0
                Ret = Api_SetWindowLong(Button(i).GethWnd, GWL_STYLE, BS_RIGHT Or Ret)
            Case 1
                Ret = Api_SetWindowLong(Button(i).GethWnd, GWL_STYLE, BS_CENTER Or Ret)
            Case 2
                Ret = Api_SetWindowLong(Button(i).GethWnd, GWL_STYLE, BS_LEFT Or Ret)
            Case 3
                Ret = Api_SetWindowLong(Button(i).GethWnd, GWL_STYLE, BS_TOP Or BS_RIGHT
Or Ret)
            Case 4
                Ret = Api_SetWindowLong(Button(i).GethWnd, GWL_STYLE, BS_TOP Or Ret)
            Case 5
                Ret = Api_SetWindowLong(Button(i).GethWnd, GWL_STYLE, BS_TOP Or BS_LEFT Or
Ret)
            Case 6
                Ret = Api_SetWindowLong(Button(i).GethWnd, GWL_STYLE, BS_BOTTOM Or
BS_RIGHT Or Ret)
            Case 7
                Ret = Api_SetWindowLong(Button(i).GethWnd, GWL_STYLE, BS_BOTTOM Or Ret)
            Case 8
                Ret = Api_SetWindowLong(Button(i).GethWnd, GWL_STYLE, BS_BOTTOM Or BS_LEFT
Or Ret)
        End Select
        Button(i).SetFocus
    Next
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

コマンドボタンの作成

CreateEllipticRgn 楕円形のリージョンを作成
CombineRgn 既存の二つの領域を結合して新しい領域を作成
SetWindowRgn 指定の領域をウィンドウ領域として設定
RGN_OR(2) リージョン同士のOR結合

PictureBoxをコマンドボタン代わりにしてみました。

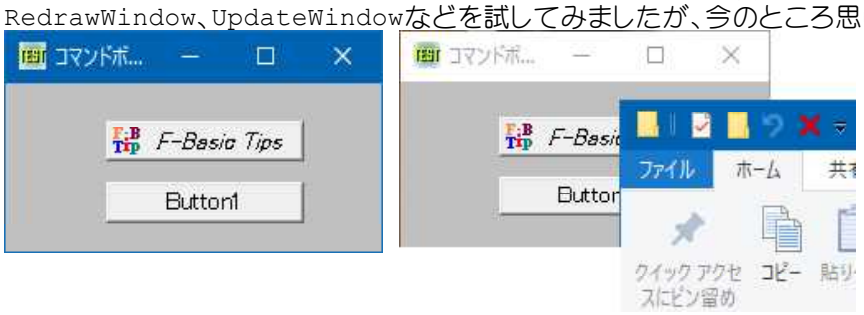

```
'=====================================
'= コマンドボタンの作成
'=    (CombineRgn2.bas)
'=====================================
#include "Windows.bi"

' 楕円形のリージョンを作成
Declare Function Api_CreateEllipticRgn& Lib "gdi32" Alias "CreateEllipticRgn" (ByVal
nLeftRect&, ByVal nTopRect&, ByVal nRightRect&, ByVal nBottomRect&)

' 既存の二つの領域を結合して新しい領域を作成
Declare Function Api_CombineRgn& Lib "gdi32" Alias "CombineRgn" (ByVal hRgnDest&, ByVal
hRgnSrc1&, ByVal hRgnSrc2&, ByVal nCombineMode&)

' 指定の領域をウィンドウ領域として設定
Declare Function Api_SetWindowRgn& Lib "user32" Alias "SetWindowRgn" (ByVal hWnd&, ByVal
hRgn&, ByVal bRedraw&)

#define RGN_OR 2                              'リージョン同士のOR結合

Var Shared Bitmap As Object
BitmapObject Bitmap

Var Shared Picture1 As Object
Var Shared Text1 As Object

Picture1.Attach GetDlgItem("Picture1")
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var Rgn0 As Long
    Var Rgn1 As Long
    Var Rgn2 As Long
    Var Rgn3 As Long
    Var Ret As Long

    Bitmap.LoadFile "Mickey1.bmp"
    Picture1.StretchBitmap Bitmap, 0, 0, 80, 80
    Bitmap.DeleteObject

    Rgn0 = Api_CreateEllipticRgn(14, 20, 68, 75)   '顔中央部
    Rgn1 = Api_CreateEllipticRgn(3, 3, 30, 33)     '左耳
    Rgn2 = Api_CreateEllipticRgn(52, 2, 77, 33)    '右耳
    Rgn3 = Api_CreateEllipticRgn(26, 33, 53, 79)   '顎

    Ret = Api_CombineRgn(Rgn0, Rgn0, Rgn1, RGN_OR) 'Rgn0 と Rgn1 を合成 = Rgn0
    Ret = Api_CombineRgn(Rgn0, Rgn0, Rgn2, RGN_OR) 'Rgn0 と Rgn2 を合成 = Rgn0
    Ret = Api_CombineRgn(Rgn0, Rgn0, Rgn3, RGN_OR) 'Rgn0 と Rgn3 を合成 = Rgn0
```

```
    Ret = Api_SetWindowRgn(Picture1.GethWnd, Rgn0, True)
    Picture1.ShowWindow -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Picture1_MouseDown edecl (ByVal Button As Integer, ByVal Shift As Integer,
ByVal x As Single, ByVal y As Single)
Sub Picture1_MouseDown(ByVal Button As Integer, ByVal Shift As Integer, ByVal x As Single,
ByVal y As Single)

    If Button = 1 Then
        Picture1.MoveWindow 9, 9
        Bitmap.LoadFile "Mickey2.bmp"
        Picture1.StretchBitmap Bitmap, 0, 0, 80, 80
        Bitmap.DeleteObject
    End If
End Sub

'=====================================
'=
'=====================================
Declare Sub Picture1_MouseUp edecl (ByVal Button As Integer, ByVal Shift As Integer, ByVal
x As Single, ByVal y As Single)
Sub Picture1_MouseUp(ByVal Button As Integer, ByVal Shift As Integer, ByVal x As Single,
ByVal y As Single)

    Picture1.MoveWindow 8, 8
    Bitmap.LoadFile "Mickey1.bmp"
    Picture1.StretchBitmap Bitmap, 0, 0, 80, 80
    Bitmap.DeleteObject

    Text1.SetWindowtext "ミッキーマウスボタンが押されました！"
    Wait 100
    Text1.SetWindowText ""
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

コマンドボタンを作成しアイコンを描画

ピクチャボックスを利用して、コマンドボタン、アイコンおよびフォーカス用点線も描画してみます。
CreateFontIndirect 論理フォントを作成
SelectObject 指定されたデバイスコンテキストのオブジェクトを選択
GetSysColor システムの背景色を取得
SetBkColor デバイスコンテキストの背景色を設定
DrawFrameControl 指定されたタイプとスタイルを備えるボタンやスクロールバー等のフレームコントロール描画
DrawText 文字列を指定領域に出力
ExtractIconEx EXE･DLLから大きいアイコン･小さいアイコンを取得
DrawIconEx アイコンを描画
DrawEdge 矩形に3D効果を与える
DrawFocusRect フォーカスを得た時の点線の枠を描く
SetCapture 指定のウィンドウにマウスキャプチャを設定
ReleaseCapture マウスのキャプチャを解放
DestroyIcon アイコンのハンドルを解放
GetDC デバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放
DeleteObject システムリソースを解放

ピクチャボックスを利用してコマンドボタンを作成しアイコンを描画しています。
あらかじめ作成してある、16ドット・32ドットサイズのアイコン「favicon.ico」、16ドットサイズのBitmapを利用しています。
他のウィンドウが重なったとき、リサイズ時に再描画されないなど欠点があります。Timerで強制的に再描画していますが、ちょっと挙動が変ですね(^^;)
Visual Basic での Refresh と、F-Basic でのそれとは、動作が異なるようです。
RedrawWindow、UpdateWindowなどを試してみましたが、今のところ思い通りの動作はしてくれません。




```
'=======================================
'= コマンドボタンとアイコン描画
'=   フォーカス用点線も描画
'=   (DrawFrameControl5.bas)
'=======================================
#include "Windows.bi"

Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

#define LF_FACESIZE 32

Type LOGFONT
    lfHeight          As Long
    lfWidth           As Long
    lfEscapement      As Long
    lfOrientation     As Long
    lfWeight          As Long
    lfItalic          As byte
    lfUnderline       As byte
    lfStrikeOut       As byte
    lfCharSet         As byte
    lfOutPrecision    As byte
    lfClipPrecision   As byte
    lfQuality         As byte
    lfPitchAndFamily  As byte
    lfFaceName(LF_FACESIZE) As byte
End Type

' 論理フォントを作成
Declare Function Api_CreateFontIndirect& Lib "gdi32" Alias "CreateFontIndirectA"
(lpLogFont As LOGFONT)

' 指定されたデバイスコンテキストのオブジェクトを選択
Declare Function Api_SelectObject& Lib "gdi32" Alias "SelectObject" (ByVal hDC&, ByVal
hObject&)

' システムの背景色を取得
Declare Function Api_GetSysColor& Lib "user32" Alias "GetSysColor" (ByVal nIndex&)

' デバイスコンテキストの背景色を設定
Declare Function Api_SetBkColor& Lib "gdi32" Alias "SetBkColor" (ByVal hDC&, ByVal
crColor&)

' 指定されたタイプとスタイルを備える、ボタンやスクロールバーなどのフレームコントロールを描画
Declare Function Api_DrawFrameControl& Lib "user32" Alias "DrawFrameControl" (ByVal
hDC&, lpRect As RECT, ByVal un1&, ByVal un2&)
```

```
' 文字列を指定領域に出力
Declare Function Api_DrawText& Lib "user32" Alias "DrawTextA" (ByVal hDC&, ByVal lpStr$,
ByVal nCount&, lpRect As RECT, ByVal wFormat&)

' EXE･DLLから大きいアイコン･小さいアイコンを取得
Declare Function Api_ExtractIconEx& Lib "shell32" Alias "ExtractIconExA" (ByVal
lpszFile$, ByVal nIconIndex&, phiconLarge&, phiconSmall&, ByVal nIcons&)

' アイコンを描画
Declare Function Api_DrawIconEx& Lib "user32" Alias "DrawIconEx" (ByVal hDC&, ByVal
xLeft&, ByVal yTop&, ByVal hIcon&, ByVal cxWidth&, ByVal cyWidth&, ByVal istepIfAniCur&,
ByVal hbrFlickerFreeDraw&, ByVal diFlags&)

' 矩形に3D効果を与える
Declare Function Api_DrawEdge& Lib "user32" Alias "DrawEdge" (ByVal hDC&, qrc As RECT,
ByVal edge&, ByVal grfFlags&)

' フォーカスを得た時の点線の枠を描く
Declare Function Api_DrawFocusRect& Lib "user32" Alias "DrawFocusRect" (ByVal hDC&,
lpRect As RECT)

' 指定のウィンドウにマウスキャプチャを設定
Declare Function Api_SetCapture& Lib "user32" Alias "SetCapture" (ByVal hWnd&)

' マウスのキャプチャを解放
Declare Function Api_ReleaseCapture& Lib "user32" Alias "ReleaseCapture" ()

' アイコンのハンドルを解放
Declare Function Api_DestroyIcon& Lib "user32" Alias "DestroyIcon" (ByVal hIcon&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

' 論理オブジェクトを削除し、そのオブジェクトに関連付けられていたすべてのシステムリソースを解放
Declare Function Api_DeleteObject& Lib "gdi32" Alias "DeleteObject" (ByVal hObject&)

#define DFC_BUTTON 4                          'ボタンコントロールを描画
#define DFCS_BUTTONPUSH &H10                  'プッシュボタン
#define DFCS_PUSHED &H200                     '押された状態
#define DT_RIGHT &H2                          'テキストを右揃え
#define DT_LEFT &H0                           'テキストを左揃え
#define DT_CENTER &H1                         'テキストを水平方向に中央揃えで表示します。
#define DT_VCENTER &H4                        'テキストを垂直方向の中央揃え。DT_SINGLELINEと同時に
                                              指定する必要がある
#define DT_SINGLELINE &H20                    'テキストを改行せず、一行で表示
#define COLOR_BTNFACE 15                      'コマンドボタンの表面色
#define LR_LOADFROMFILE &H10                  '外部ファイルからロードする
#define DST_BITMAP &H4                        'ビットマップ
#define DI_NORMAL 3                           'DI_IMAGEとDI_MASKの組み合わせ
#define BF_RECT (&H1 Or &H2 Or &H4 Or &H8)    '(BF_LEFT Or BF_TOP Or BF_RIGHT Or BF_BOTTOM)

#define RDW_ALLCHILDREN &H80                  '子ウィンドウがあるときは、それらも再描画の対象とする
#define RDW_ERASE &H4                         'ウィンドウの再描画時に、WM_ERASEBKGNDを送る。
                                              RDW_INVALIDATEフラグと同時に指定する必要がある
#define RDW_ERASENOW &H200                    '必要であれば、関数が制御を返す前に、影響を受けたウィン
                                              ドウ(RDW_ALLCHILDRENまたはRDW_NOCHILDREN
#define RDW_FRAME &H400                       '更新領域と交差する非クライアント領域の部分にも、
                                              WM_NCPAINTを送る。RDW_INVALIDATEフラグと同時に指
                                              定する
#define RDW_INTERNALPAINT &H2                 'ウィンドウが無効な領域を持つか持たないかに関わらず、
                                              WM_PAINTメッセージをウィンドウにポストする
#define RDW_INVALIDATE &H2                    '指定したウィンドウ内の領域を無効化する
#define RDW_NOCHILDREN &H40                   '子ウィンドウがある時は、それらを再描画の対象から外す
#define RDW_NOERASE &H20                      '未処理 WM_ERASEBKGNDを送らないようにする
#define RDW_NOFRAME &H800                     '未処理のWM_NCPAINTを送らないようにする。
                                              RDW_VALIDATEフラグと同時に指定する必要がる
#define RDW_NOINTERNALPAINT &H10              '未処理の内部 WM_PAINTを送らないようにします。
```

```
#define RDW_UPDATENOW &H100                 '必要であれば、関数が制御を返す前に、影響を受けたウィンドウ(RDW_ALLCHILDRENフラグまたはRDW_NOCHILDREN
#define RDW_VALIDATE &H8                    '指定したウィンドウ内の領域を有効化する

Var Shared mPushed As Integer
Var Shared mFocus As Integer
Var Shared rc As RECT
Var Shared Picture1 As Object
Var Shared Button1 As Object
Var Shared Timer1 As Object

Picture1.Attach GetDlgItem("Picture1")
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Timer1.Attach GetDlgItem("Timer1")

'=====================================
'= コマンドボタンを描画
'=====================================
Declare Sub ButtonDraw edecl ()
Sub ButtonDraw()
    Var rc2 As RECT
    Var hDC As Long
    Var hIcon As Long
    Var lIcon As Long
    Var sIcon As Long
    Var State As Long
    Var Caption As String
    Var lf As LOGFONT
    Var rFont As Long
    Var Size As Integer
    Var BkCol As Long
    Var Edge As Long
    Var Ret As Long

    'Picture1のデバイスコンテキストを取得
    hDC = Api_GetDC(Picture1.GethWnd)

    'ハンドルを取得
    'Iconの場合(32x32、16x16の合成)
    hIcon = Api_ExtractIconEx("favicon.ico", 0, lIcon, sIcon, 1)

    'Bmpの場合(16x16)
'   hIcon = Api_ExtractIconEx("fbtip16.bmp", 0, lIcon, sIcon, 1)

    'キャプションを設定
    Caption = "    F-Basic Tips"

    'ボタンの色をシステムより取得
    BkCol = Api_GetSysColor(COLOR_BTNFACE)

    'バックカラーに指定
    Ret = Api_SetBkColor(hDC, BkCol)

    'フォントサイズ、日本語、イタリック指定
    Size = 14
    lf.lfHeight = Size
    lf.lfCharSet = 128
    lf.lfItalic = 1
    rFont = Api_CreateFontIndirect(lf)
    Ret = Api_SelectObject(hDC, rFont)

    'ボタンの凹凸を設定
    State = DFCS_BUTTONPUSH
    If mPushed = True Then
        State = State Or DFCS_PUSHED
    End If

    'ボタンの矩形領域を指定
    rc.Left = 0 : x = 0
    rc.Top = 0 : y = 0
```

```
    rc.Right = Picture1.GetWidth
    rc.Bottom = Picture1.GetHeight

    'フォーカス用の矩形領域を指定
    rc2.Left = rc.Left + 4
    rc2.Top = rc.Top + 4
    rc2.Right = rc.Right - 4
    rc2.Bottom = rc.Bottom - 4

    'ボタンを描画
    Ret = Api_DrawFrameControl(hDC, rc, DFC_BUTTON, State)

    '押された場合、キャプションをずらす
    If mPushed = True Then
        rc.Left = 2 : x = 2
        rc.Top = 2 : y = 2
    End If

    'ボタンの外形
    Ret = Api_DrawEdge(hDC, rc, Edge, BF_RECT)

    'キャプションを描画
    Ret = Api_DrawText(hDC, Caption, Len(Caption), rc, DT_CENTER Or DT_VCENTER Or
DT_SINGLELINE)

    'Iconを描画
    Ret = Api_DrawIconEx(hDC, x + 4, y + 4, sIcon, 0, 0, ByVal 0, 0, DI_NORMAL)

    'フォーカス用の矩形を描く
    If mFocus = True Then
        Ret = Api_DrawFocusRect(hDC, rc2)
    End If

    'デバイスコンテキスト、ハンドルの解放
    Ret = Api_ReleaseDC(Picture1.GethWnd, hDC)
    Ret = Api_DestroyIcon(hIcon)
    Ret = Api_DeleteObject(rFont)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Timer1.SetInterval 5
    Timer1.Enable -1
End Sub

'=====================================
'= マウスを動かしたとき
'=====================================
Declare Sub Picture1_MouseMove edecl(ByVal Button As Integer, ByVal Shift As Integer,
ByVal x As Single, ByVal y As Single)
Sub Picture1_MouseMove(ByVal Button As Integer, ByVal Shift As Integer, ByVal x As Single,
ByVal y As Single)
    Var Ret As Long

    'マウスが Picture 上にあるとき
    If x >= 0 And y >= 0 And x <= Picture1.GetWidth And y <= Picture1.GetHeight Then
        Ret = Api_SetCapture(Picture1.GethWnd)
        ButtonDraw

    'マウスが Picture1 を外れたとき
    Else
        Ret = Api_ReleaseCapture
        mPushed = False
        mFocus = False
        ButtonDraw
    End If
End Sub
```

```
'=====================================
'= マウスを押したとき
'=====================================
Declare Sub Picture1_MouseDown edecl (Button As Integer, Shift As Integer, x As Single, y
As Single)
Sub Picture1_MouseDown(Button As Integer, Shift As Integer, x As Single, y As Single)
    Var Ret As Long

    Ret = Api_SetCapture(Picture1.GethWnd)
    mPushed = True
    mFocus = True
    ButtonDraw
    SetFocus
End Sub

'=====================================
'= マウスを離したとき
'=====================================
Declare Sub Picture1_MouseUp edecl (Button As Integer, Shift As Integer, x As Single, y As
Single)
Sub Picture1_MouseUp(Button As Integer, Shift As Integer, x As Single, y As Single)
    Var Ret As Long

    Ret = Api_ReleaseCapture
    mPushed = False
    ButtonDraw
End Sub

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    ButtonDraw
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

コマンドボタンを描画

ピクチャボックスを利用し、コマンドボタンを作成します。
CreateFontIndirect 論理フォントを作成
SelectObject 指定されたデバイスコンテキストのオブジェクトを選択
GetSysColor システムの背景色を取得
SetBkColor デバイスコンテキストの背景色を設定
DrawFrameControl 指定されたタイプとスタイルを備える、ボタンやスクロールバー等のフレームコントロール描画
DrawText 文字列を指定領域に出力
GetDC ディスプレイデバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放

```
'====================================
'= コマンドボタンを描画
'=   フォーカス点線無し
'=   (DrawFrameControl4.bas)
'====================================
#include "Windows.bi"

Type RECT
    Left       As Long
    Top        As Long
    Right      As Long
    Bottom     As Long
End Type

#define LF_FACESIZE 32

Type LOGFONT
    lfHeight          As Long
    lfWidth           As Long
    lfEscapement      As Long
    lfOrientation     As Long
    lfWeight          As Long
    lfItalic          As byte
    lfUnderline       As byte
    lfStrikeOut       As byte
    lfCharSet         As byte
    lfOutPrecision    As byte
    lfClipPrecision   As byte
    lfQuality         As byte
    lfPitchAndFamily As byte
    lfFaceName(LF_FACESIZE) As byte
End Type

' 論理フォントを作成
Declare Function Api_CreateFontIndirect& Lib "gdi32" Alias "CreateFontIndirectA"
(lpLogFont As LOGFONT)

' 指定されたデバイスコンテキストのオブジェクトを選択
Declare Function Api_SelectObject& Lib "gdi32" Alias "SelectObject" (ByVal hDC&, ByVal
hObject&)

' システムの背景色を取得
Declare Function Api_GetSysColor& Lib "user32" Alias "GetSysColor" (ByVal nIndex&)

' デバイスコンテキストの背景色を設定
Declare Function Api_SetBkColor& Lib "gdi32" Alias "SetBkColor" (ByVal hDC&, ByVal
crColor&)

' 指定されたタイプとスタイルを備える、ボタンやスクロールバーなどのフレームコントロールを描画
Declare Function Api_DrawFrameControl& Lib "user32" Alias "DrawFrameControl" (ByVal
hDC&, lpRect As RECT, ByVal un1&, ByVal un2&)

' 文字列を指定領域に出力
Declare Function Api_DrawText& Lib "user32" Alias "DrawTextA" (ByVal hDC&, ByVal lpStr$,
ByVal nCount&, lpRect As RECT, ByVal wFormat&)

' 指定のウィンドウにマウスキャプチャを設定
Declare Function Api_SetCapture& Lib "user32" Alias "SetCapture" (ByVal hWnd&)

' マウスのキャプチャを解放
Declare Function Api_ReleaseCapture& Lib "user32" Alias "ReleaseCapture" ()

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

#define DT_CENTER &H1                          'テキストを水平方向に中央揃えで表示します。
#define DT_VCENTER &H4                         'テキストを垂直方向の中央揃え。DT_SINGLELINEと同時に
```

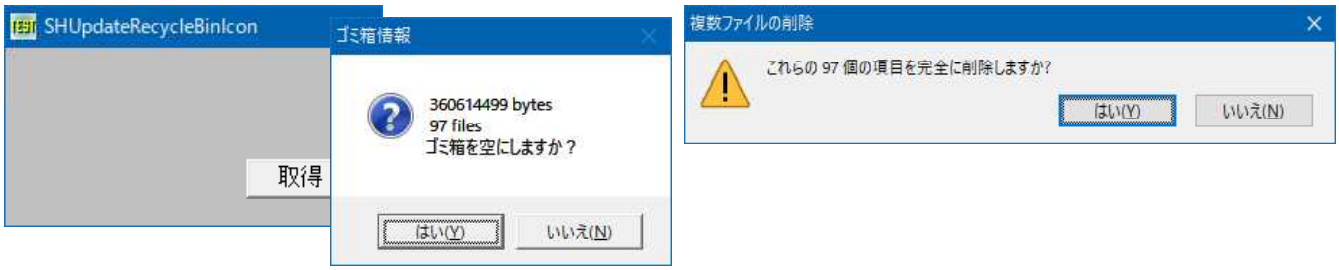
```
                                        指定する必要がある
#define DT_SINGLELINE &H20              'テキストを改行せず、一行で表示
#define BDR_RAISEDOUTER &H1             '外側エッジが凸
#define BDR_SUNKENOUTER &H2             '外側エッジが凹
#define BDR_RAISEDINNER &H4             '内側が凸
#define BDR_SUNKENINNER &H8             '内側が凹
#define BDR_OUTER &H3                   '
#define BDR_INNER &HC                   '
#define BF_RECT &HF                     '(BF_LEFT Or BF_TOP Or BF_RIGHT Or BF_BOTTOM)
#define DFC_BUTTON 4                    'ボタンコントロールを描画
#define DFCS_BUTTONPUSH &H10            'プッシュボタン
#define DFCS_PUSHED &H200               '押された状態
#define COLOR_BTNFACE 15                'コマンドボタンの表面色

Var Shared Pushed As Integer

Var Shared Picture1 As Object
Picture1.Attach GetDlgItem("Picture1")

'=====================================
'=
'=====================================
Declare Sub ButtonDraw edecl ()
Sub ButtonDraw()
    Var rc As RECT
    Var lf As LOGFONT
    Var hDC As Long
    Var State As Long
    Var Caption As String
    Var rFont As Long
    Var BkCol As Long

    'Picture1のハンドルを取得
    hDC = Api_GetDC(Picture1.GethWnd)

    'ボタンの色をシステムより取得
    BkCol = Api_GetSysColor(COLOR_BTNFACE)

    'バックカラーに指定
    Ret = Api_SetBkColor(hDC, BkCol)

    'ボタンの凹凸を設定
    State = DFCS_BUTTONPUSH
    If Pushed = True Then
        State = State Or DFCS_PUSHED
    End If

    'キャプションを設定
    Caption = "ButtonEx"

    'フォントサイズ14P、日本語、イタリック指定
    lf.lfHeight = 14
    lf.lfCharSet = 128
    lf.lfItalic = 1
    rFont = Api_CreateFontIndirect(lf)
    Ret = Api_SelectObject(hDC, rFont)

    'ボタンのサイズを設定
    rc.Left = 0
    rc.Top = 0
    rc.Right = Picture1.GetWidth
    rc.Bottom = Picture1.GetHeight

    'ボタンを描画
    Ret = Api_DrawFrameControl(hDC, rc, DFC_BUTTON, State)

    '押されている状態の時は2ドットずらす
    If Pushed = True Then
        rc.Left = 2
        rc.Top = 2
```

```
    End If

    'ボタンのテキストの表示
    Ret = Api_DrawText(hDC, Caption, Len(Caption), rc, DT_CENTER Or DT_VCENTER Or
DT_SINGLELINE)

    'デバイスコンテキストの解放
    Ret = Api_ReleaseDC(Picture1.GethWnd, hDC)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Ret = Api_SetCapture(Picture1.GethWnd)
    ButtonDraw
End Sub

'=====================================
'= マウスを動かしたとき
'=====================================
Declare Sub Picture1_MouseMove edecl(ByVal Button As Integer, ByVal Shift As Integer,
ByVal x As Single, ByVal y As Single)
Sub Picture1_MouseMove(ByVal Button As Integer, ByVal Shift As Integer, ByVal x As Single,
ByVal y As Single)
    Var Ret As Long

    'マウスが Picture 上にあるとき
    If x >= 0 And y >= 0 And x <= Picture1.GetWidth And y <= Picture1.GetHeight Then
        Ret = Api_SetCapture(Picture1.GethWnd)
        ButtonDraw

    'マウスが Picture1 を外れたとき
    Else
        Ret = Api_ReleaseCapture
        Pushed = False
        ButtonDraw
    End If
End Sub

'=====================================
'= マウスを押したとき
'=====================================
Declare Sub Picture1_MouseDown edecl (Button As Integer, Shift As Integer, x As Single, y
As Single)
Sub Picture1_MouseDown(Button As Integer, Shift As Integer, x As Single, y As Single)
    Var Ret As Long

    Ret = Api_SetCapture(Picture1.GethWnd)
    Pushed = True
    ButtonDraw
    SetFocus
End Sub

'=====================================
'= マウスを離したとき
'=====================================
Declare Sub Picture1_MouseUp edecl (Button As Integer, Shift As Integer, x As Single, y As
Single)
Sub Picture1_MouseUp(Button As Integer, Shift As Integer, x As Single, y As Single)
    Var Ret As Long

    Ret = Api_ReleaseCapture
    Pushed = False
    ButtonDraw
End Sub
```

```
'====================================
'=
'====================================
While 1
    WaitEvent
Wend
Stop
End
```

ゴミ箱のサイズ取得と削除

SHQueryRecycleBin ごみ箱のサイズとアイテム数
SHUpdateRecycleBinIcon ゴミ箱のアイコンをアップデート
SHEmptyRecycleBin 指定のドライブのごみ箱を空にする
MoveMemory メモリの指定領域をコピー



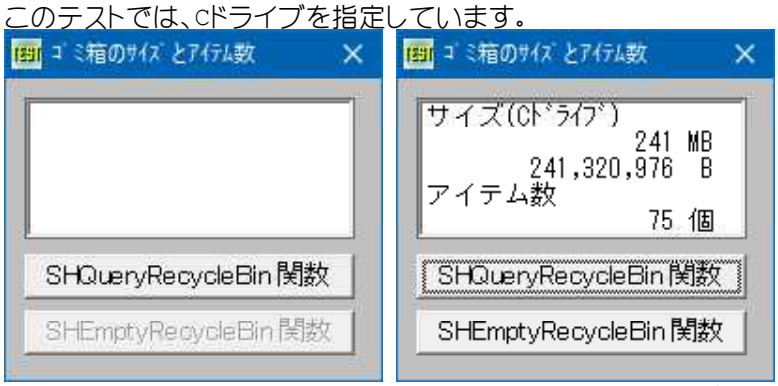

```
'====================================
'= ゴミ箱のサイズ取得と削除
'=    (SHUpdateRecycleBinIcon.bas)
'====================================
#include "Windows.bi"

Type ULARGE_INTEGER
    LowPart    As Long
    HighPart   As Long
End Type

Type SHQUERYRBINFO
    cbSize           As Long
    i64Size          As ULARGE_INTEGER
    i64NumItems      As ULARGE_INTEGER
End Type

' ごみ箱のサイズとアイテム数
Declare Function Api_SHQueryRecycleBin& Lib "shell32" Alias "SHQueryRecycleBinA" (ByVal
pszRootPath$, pSHQueryRBInfo As SHQUERYRBINFO)

' ゴミ箱のアイコンをアップデート
Declare Function Api_SHUpdateRecycleBinIcon& Lib "shell32" Alias
"SHUpdateRecycleBinIcon" ()

' 指定のドライブのごみ箱を空にする
Declare Function Api_SHEmptyRecycleBin& Lib "shell32" Alias "SHEmptyRecycleBinA" (ByVal
hWnd&, ByVal pszRootPath$, ByVal dwFlags&)

' メモリの指定領域をコピー
Declare Sub MoveMemory Lib "Kernel32" Alias "RtlMoveMemory" (Dest As Any, Source As Any,
ByVal length&)

#define SHERB_NOCONFIRMATION &H1          '削除の確認をしない
#define SHERB_NOPROGRESSUI &H2            '進行状況を表示しない
#define SHERB_NOSOUND &H4                 '処理完了時にサウンド無し
#define GENERIC_READ -2147483648          '読み込みモード(&H80000000)
#define vbNullString ByVal 0              '値0の文字列。値0を持つ文字列。空文字列ではない

Var Shared Button1 As Object
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
```

```
'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var si As SHQUERYRBINFO
    Var Ret As Long

    si.cbSize = Len(si)
    Ret = Api_SHQueryRecycleBin(vbNullString, si)

    If (si.i64Size.LowPart And GENERIC_READ) = GENERIC_READ Or si.i64Size.HighPart > 0
Then
        A% = MessageBox("ゴミ箱情報", "2GBを越えています" & Chr$(13, 10) &  "ゴミ箱を空にします
か？", 4, 1)
    Else
        A% = MessageBox("ゴミ箱情報", Str$(si.i64Size.LowPart) & " bytes" & Chr$(13, 10) &
Str$(si.i64NumItems.LowPart) & " files" & Chr$(13, 10) & "ゴミ箱を空にしますか？", 4, 1)
    End If

    If A% = 5 Then
        Ret = Api_SHEmptyRecycleBin(GethWnd, vbNullString, 0)
        Ret = Api_SHUpdateRecycleBinIcon
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

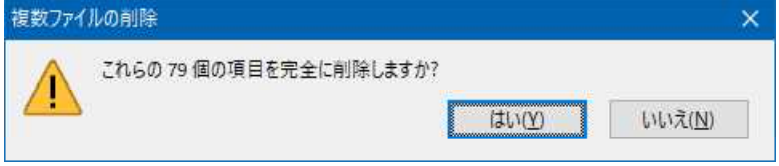
ゴミ箱のサイズとアイテム数の取得

ゴミ箱のサイズとアイテム数を取得します。
SHQueryRecycleBin ゴミ箱のサイズとアイテム数を取得
SHEmptyRecycleBin 指定のドライブのごみ箱を空にする

このテストでは、cドライブを指定しています。



SHEmptyRecycleBin関数を実行すると下記確認メッセージが表示されます。

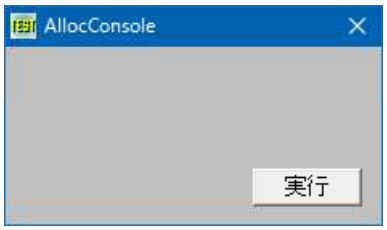


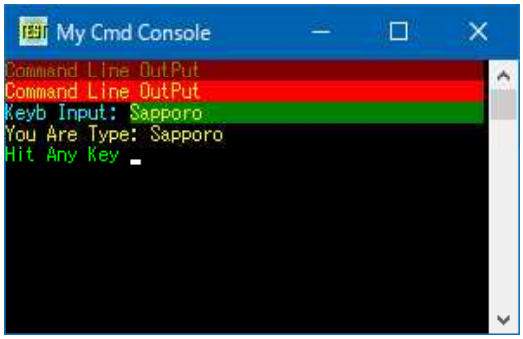
```
'=====================================
'= ゴミ箱のサイズとアイテム数の取得
'=    (SHQueryRecycleBin.bas)
'=====================================
#include "Windows.bi"

Type SHQUERYRBINFO
    cbSize            As Long              '構造体のバイト数
    i64SizeLow        As Long              'ごみ箱にある全アイテム数の下位32ビット値
    i64SizeHigh       As Long              '同、上位32ビット値
    i64NumItemsLow    As Long              'ごみ箱にあるアイテム数の下位32ビット値
    i64NumItemsHigh   As Long              '同、上位32ビット値
End Type

' ごみ箱のサイズとアイテム数
Declare Function Api_SHQueryRecycleBin& Lib "Shell32" Alias "SHQueryRecycleBinA" (ByVal
pszRootPath$, pSHQueryRBInfo As SHQUERYRBINFO)

' 指定のドライブのごみ箱を空にする
Declare Function Api_SHEmptyRecycleBin& Lib "Shell32" Alias "SHEmptyRecycleBinA" (ByVal
hWnd&, ByVal pszRootPath$, ByVal dwFlags&)

' dwFlagsの定数
#define SHERB_NOCONFIRMATION &H1           '削除の確認をしない
#define SHERB_NOPROGRESSUI &H2             '進行状況を表示しない
#define SHERB_NOSOUND &H4                  '処理完了時にサウンド無し

Var Shared List1 As Object
Var Shared Button2 As Object
List1.Attach GetDlgItem("List1")
Button2.Attach GetDlgItem("Button2")

'=====================================
'=
'=====================================
Declare Sub Mainform_Start edecl ()
Sub Mainform_Start()
    Button2.EnableWindow 0
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var pSHQueryRBInfo As SHQUERYRBINFO
    Var psize$ As String
    Var pitem$ As String
    Var Ret As Long

    pSHQueryRBInfo.cbSize = Len(pSHQueryRBInfo)

    '関数の実行
    Ret = Api_SHQueryRecycleBin("C:¥", pSHQueryRBInfo)

    '結果表示
    psize$ = Str$(pSHQueryRBInfo.i64SizeLow)
    psize$ = Str$(pSHQueryRBInfo.i64SizeHigh) + Mid$(psize$, 2)

    pitem$ = Str$(pSHQueryRBInfo.i64NumItemsLow)
    pitem$ = Str$(pSHQueryRBInfo.i64NumItemsHigh) + Mid$(pitem$, 2)

    List1.ResetContent
    List1.AddString "サイズ(Cﾄﾞﾗｲﾌﾞ)"

    List1.AddString space$(4) + Format$(Int(Val(psize$) / (10 ^ 6)), "###,###,###,###
MB")
    List1.AddString space$(4) + Format$(Int(Val(psize$))          , "###,###,###,### B")

    List1.AddString "アイテム数"
```

```
    List1.AddString space$(4) + Format$(Val(pitem$)                , "       ###,### 個")

    If val(pitem$) > 0 Then
        Button2.EnableWindow -1
    Else
        Button2.EnableWindow 0
    End If
End Sub

'=======================================
'=
'=======================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Ret As Long

    Ret = Api_SHEmptyRecycleBin(GethWnd, "C:¥", 0)
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

コンソールウィンドウの操作(32bit○、64bit○)

AllocConsole 呼び出し側プロセスに新しいコンソールを割り当てる
FreeConsole 呼び出し側プロセスをそのコンソールから切り離す
CloseHandle オープンされているオブジェクトハンドルをクローズ
GetStdHandle 標準入力デバイス、標準出力デバイス、標準エラーデバイスのいずれかのハンドルを返す
WriteConsole コンソールスクリーンバッファの現在のカーソル位置に文字列を挿入
ReadConsole コンソール入力バッファから文字入力を読み取り、読み取った文字をバッファから削除
SetConsoleTextAttribute スクリーンバッファに書き込む文字、エコーする文字の前景(テキスト)色属性と背景色属性を設定
SetConsoleTitle コンソールウィンドウのタイトル文字列を設定






```
'=======================================
'= コンソールウィンドウの操作
'=    (AllocConsole3.bas)
'=======================================
#include "Windows.bi"

#define FOREGROUND_BLUE &H1                 '前景色(青)
#define FOREGROUND_GREEN &H2                '前景色(緑)
#define FOREGROUND_RED &H4                  '前景色(赤)
#define FOREGROUND_INTENSITY &H8            '前景色(強調)
#define FOREGROUND_SEARCH &H10              '
#define BACKGROUND_BLUE &H10                '背景色(青)
#define BACKGROUND_GREEN &H20               '背景色(緑)
#define BACKGROUND_RED &H40                 '背景色(赤)
```

```
#define BACKGROUND_INTENSITY &H80          '背景色(強調)
#define BACKGROUND_SEARCH &H20             '
#define ENABLE_LINE_INPUT &H2
#define ENABLE_ECHO_INPUT &H4
#define ENABLE_MOUSE_INPUT &H10
#define ENABLE_PROCESSED_INPUT &H1
#define ENABLE_WINDOW_INPUT &H8
#define ENABLE_PROCESSED_OUTPUT &H1
#define ENABLE_WRAP_AT_EOL_OUTPUT &H2
#define STD_OUTPUT_HANDLE (-11)
#define STD_INPUT_HANDLE (-10)
#define STD_ERROR_HANDLE (-12)
#define INVALID_HANDLE_VALUE (-1)

' 呼び出し側プロセスに新しいコンソールを割り当てる
Declare Function Api_AllocConsole& Lib "kernel32" Alias "AllocConsole" ()

' 呼び出し側プロセスをそのコンソールから切り離す
Declare Function Api_FreeConsole& Lib "kernel32" Alias "FreeConsole" ()

' オープンされているオブジェクトハンドルをクローズ
Declare Function Api_CloseHandle& Lib "Kernel32" Alias "CloseHandle" (ByVal hObject&)

' 標準入力デバイス、標準出力デバイス、標準エラーデバイスのいずれかのハンドルを返す
Declare Function Api_GetStdHandle& Lib "kernel32" Alias "GetStdHandle" (ByVal
nStdHandle&)

' コンソールスクリーンバッファの現在のカーソル位置に文字列を挿入
Declare Function Api_WriteConsole& Lib "kernel32" Alias "WriteConsoleA" (ByVal
hConsoleOutput&, lpBuffer As Any, ByVal nNumberOfCharsToWrite&,
lpNumberOfCharsWritten&, lpReserved As Any)

' コンソール入力バッファから文字入力を読み取り、読み取った文字をバッファから削除
Declare Function Api_ReadConsole& Lib "kernel32" Alias "ReadConsoleA" (ByVal
hConsoleInput&, ByVal lpBuffer$, ByVal nNumberOfCharsToRead&, lpNumberOfCharsRead&,
lpReserved As Any)

' スクリーンバッファに書き込む文字、エコーする文字の前景(テキスト)色属性と背景色属性を設定
Declare Function Api_SetConsoleTextAttribute& Lib "kernel32" Alias
"SetConsoleTextAttribute" (ByVal hConsoleOutput&, ByVal wAttributes&)

' コンソールウィンドウのタイトル文字列を設定
Declare Function Api_SetConsoleTitle& Lib "kernel32" Alias "SetConsoleTitleA" (ByVal
lpConsoleTitle$)

Var Shared hCmdIn As Long
Var Shared hCmdOut As Long
Var Shared hCmdError As Long

Var Shared MainForm As Object
Var Shared Button1 As Object

MainForm.Attach GetDlgItem("MainForm")
Button1.Attach GetDlgItem("Button1") : Button1.setFontSize 14

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var CmdIn As String * 256
    Var CmdOUT As String
    Var Ret As Long

    'コンソールウィンドウをオープン
    Ret = Api_AllocConsole

    'コンソールにタイトルを設定
    Ret = Api_SetConsoleTitle("My Cmd Console")
```

```
    'コンソールのハンドルを取得
    hCmdIn = Api_GetStdHandle(STD_INPUT_HANDLE)
    hCmdOut = Api_GetStdHandle(STD_OUTPUT_HANDLE)
    hCmdError = Api_GetStdHandle(STD_ERROR_HANDLE)

    '文字列
    CmdOUT = "Command Line OutPut" & Chr$(13, 10)

    '背景色を暗い赤にに、文字列を暗い黄に設定
    Ret = Api_SetConsoleTextAttribute(hCmdOut, FOREGROUND_RED Or FOREGROUND_GREEN Or
BACKGROUND_RED)

    'コンソールに文字列を書き込み
    Ret = Api_WriteConsole(hCmdOut, CmdOUT, Len(CmdOUT), 1, 1)

    '背景色を赤(FOREGROUND_INTENSITY)に、文字列を黄(BACKGROUND_INTENSITY)に設定
    Ret = Api_SetConsoleTextAttribute(hCmdOut, FOREGROUND_RED Or FOREGROUND_GREEN Or
FOREGROUND_INTENSITY Or BACKGROUND_RED Or BACKGROUND_INTENSITY)

    'コンソールに文字列を書き込み
    Ret = Api_WriteConsole(hCmdOut, CmdOUT, Len(CmdOUT), 1, 1)

    '文字列
    CmdOUT = "Keyb Input: "

    '背景色を黒に、文字列を薄緑に設定
    Ret = Api_SetConsoleTextAttribute(hCmdOut, FOREGROUND_BLUE Or FOREGROUND_GREEN Or
FOREGROUND_INTENSITY)

    'コンソールに文字列を書き込み
    Ret = Api_WriteConsole(hCmdOut, CmdOUT, Len(CmdOUT), 1, 1)

    '背景色を暗い緑に、文字列を黄に設定
    Ret = Api_SetConsoleTextAttribute(hCmdOut, FOREGROUND_RED Or FOREGROUND_GREEN Or
FOREGROUND_INTENSITY OR BACKGROUND_GREEN)

    'コンソールから文字列を読み込み
    Ret = Api_ReadConsole(hCmdIn, CmdIn, Len(CmdIn), 1, 1)

    CmdOUT = "You Are Type: " & Left$(CmdIn, InStr(CmdIn, Chr$(0)) - 3) & Chr$(13, 10)

    '背景色を暗い緑に、文字列を黄に設定
    Ret = Api_SetConsoleTextAttribute(hCmdOut, FOREGROUND_RED Or FOREGROUND_GREEN Or
FOREGROUND_INTENSITY)

    'コンソールに文字列を書き込み
    Ret = Api_WriteConsole(hCmdOut, CmdOUT, Len(CmdOUT), 1, 1)

    '文字列
    CmdOUT = "Hit Any Key "

    '背景色を黒に、文字列を緑に設定
    Ret = Api_SetConsoleTextAttribute(hCmdOut, FOREGROUND_GREEN Or
FOREGROUND_INTENSITY)

    'コンソールに文字列を書き込み
    Ret = Api_WriteConsole(hCmdOut, CmdOUT, Len(CmdOUT), 1, 1)

    'コンソールから文字列を読み込み
    Ret = Api_ReadConsole(hCmdIn, CmdIn, Len(CmdIn), 1, 1)

    'コンソールウィンドウを閉じる
    Ret = Api_FreeConsole

    MainForm.SetFocus
End Sub

'=====================================
'=
'=====================================
```

```
While 1
    WaitEvent
Wend
Stop
End
```

コントロール上のカーソルを変更

例では、Button上のカーソルを変更しています。
LoadCursorFromFile カーソルをファイルから取得
SetClassLong クラスに関連づけされている補足領域にlong値を設定

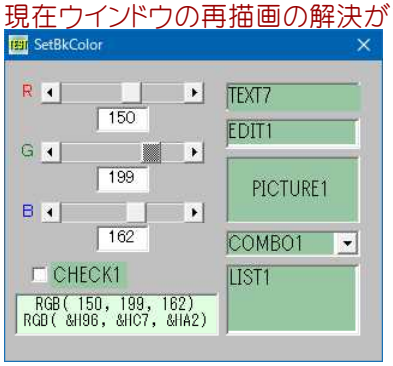

```
'=====================================
'= コントロール上のカーソルを変更
'=    (SetClassLong2.bas)
'=====================================
#include "Windows.bi"

' カーソルをファイルから取得
Declare Function Api_LoadCursorFromFile& Lib "user32" Alias "LoadCursorFromFileA" (ByVal
lpFileName$)

' クラスに関連付けている補足データ域にlong値を設定
Declare Function Api_SetClassLong& Lib "user32" Alias "SetClassLongA" (ByVal hWnd&,
ByVal nIndex&, ByVal dwNewLong&)

#define GCW_HCURSOR (-12)                    '初期値のカーソルハンド

Var Shared Button1 As Object
Button1.Attach GetDlgItem("Button1")

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var hWnd As Long
    Var Ret As Long

    'ファイルからカーソルを読み込む
    hWnd = Api_LoadCursorFromFile("C:¥Windows¥Cursors¥dinosau2.ani")

    'ファイルからカーソルを読み込む(Windows10 には上記カーソルはありません)
'   hWnd = Api_LoadCursorFromFile("C:¥Windows¥Cursors¥aero_working.ani")

    'Button上のカーソル
    Ret = Api_SetClassLong(Button1.GethWnd, GCW_HCURSOR, hWnd)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

コントロールの背景色設定テスト

コントロール(TextBox、EditBox、PictureBox)の背景色を設定するテストです。
FillRect 長方形を塗りつぶす
CreateSolidBrush 純色論理ブラシの作成
SetBKMode 背景色の塗り潰しモード設定
SetBKColor デバイスコンテキストの背景色を設定
TextOut テキストを指定の位置に出力
GetDC デバイスコンテキストの取得
ReleaseDC デバイスコンテキストの解放

スクロールバーでRGBの値を設定し、その色を各コントロールの背景色にしています。
そのときのRGB値をテキストボックスに表示しています。

現在ウインドウの再描画の解決ができていません！ エディットボックスに文字入力する際背景色が戻ってしまいます。




```
'=====================================
'= 背景色を変更する
'=    (SetBkColor.bas)
'=====================================
#include "Windows.bi"

Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

' ブラシで矩形領域を塗りつぶす
Declare Function Api_FillRect& Lib "user32" Alias "FillRect" (ByVal hDC&, ByRef r As RECT,
ByVal hBrush&)

' 純色で論理ブラシを作成
Declare Function Api_CreateSolidBrush& Lib "gdi32" Alias "CreateSolidBrush" (ByVal
crColor&)

' バックグラウンドの塗りつぶしモード設定
Declare Function Api_SetBkMode& Lib "gdi32" Alias "SetBkMode" (ByVal hDC&, ByVal
iBkMode&)

' デバイスコンテキストの背景色を設定
Declare Function Api_SetBkColor& Lib "gdi32" Alias "SetBkColor" (ByVal hDC&, ByVal
crColor&)
```

```
' デバイスコンテキストの背景色を取得(ここでは使っていません)
Declare Function Api_GetBkColor& Lib "gdi32" Alias "GetBkColor" (ByVal hDC&)

' テキストを指定の位置に出力
Declare Function Api_TextOut& Lib "gdi32" Alias "TextOutA" (ByVal hDC&, ByVal nXStart&,
ByVal nYStart&, ByVal lpString$, ByVal cbString&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

#define OPAQUE 2                                    '背景色を設定する
#define TRANSPARENT 1                               '背景色を設定しない

Var Shared Text(7) As Object
Var Shared Edit1 As Object
Var Shared Combo1 As Object
Var Shared List1 As Object
Var Shared Check1 As Object
Var Shared Picture1 As Object
Var Shared HScroll1 As Object
Var Shared HScroll2 As Object
Var Shared HScroll3 As Object

For i = 0 To 7
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1)))
    Text(i).SetFontSize 14
Next
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
Check1.Attach GetDlgItem("Check1") : Check1.SetFontSize 14
Picture1.Attach GetDlgItem("Picture1")
HScroll1.Attach GetDlgItem("HScroll1")
HScroll2.Attach GetDlgItem("HScroll2")
HScroll3.Attach GetDlgItem("HScroll3")

Var Shared hDCT As Long                             'Text(6)のデバイスコンテキスト
Var Shared hDCE As Long                             'Edit1のデバイスコンテキスト
Var Shared hDCP As Long                             'Picture1のデバイスコンテキスト
Var Shared hDCC As Long                             'Combo1のデバイスコンテキスト
Var Shared hDCL As Long                             'List1のデバイスコンテキスト
Var Shared hDCK As Long                             'Check1のデバイスコンテキスト
Var Shared rgbColor As Long
Var Shared r As Byte
Var Shared g As Byte
Var Shared b As Byte

'=====================================
'=
'=====================================
Declare Sub BackColorSet edecl ()
Sub BackColorSet()
    Var rct As RECT
    Var x As Integer
    Var y As Integer
    Var txt As String
    Var cbStr As Long
    Var colBrush As Long
    Var Ret As Long

    colBrush = Api_CreateSolidBrush(rgbColor)      '描画色を設定

    rct.Top = 0
    rct.Left = 0
    rct.Right = Text(6).GetWidth - 2
    rct.Bottom = Text(6).GetHeight - 2
```

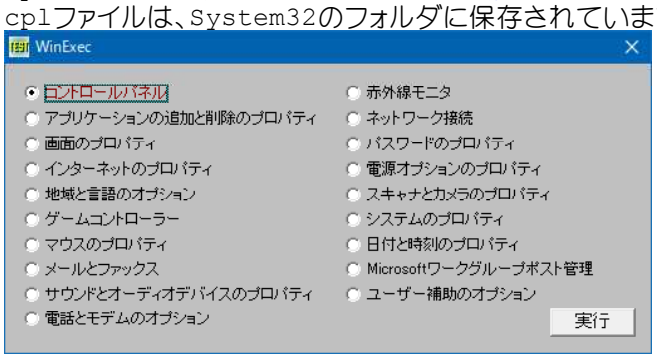
```
    txt = "TEXT7"
    cbStr = Len(txt)
    Ret = Api_FillRect(hDCT, rct, colBrush)         '長方形を設定
    Ret = Api_SetBkMode(hDCT, OPAQUE)               'Text(6)の背景色のモードを設定する
    Ret = Api_SetBkColor(hDCT, rgbColor)            '背景色を設定する
    Ret = Api_TextOut(hDCT, 0, 0, txt, cbStr)       '指定位置にtxt表示

    rct.Top = 0
    rct.Left = 0
    rct.Right = Edit1.GetWidth - 6                  '3D枠分減算
    rct.Bottom = Edit1.GetHeight  -6                '〃

    If Edit1.GetWindowText = "" Then
        txt = "EDIT1"
    Else
        txt = Edit1.GetWindowText
    End If

    cbStr = Len(txt)
    Ret = Api_FillRect(hDCE, rct, colBrush)         '長方形を設定
'   Ret = Api_SetBkMode(hDCE, OPAQUE)               'Edit1の背景色のモードを設定する
    Ret = Api_SetBkColor(hDCE, rgbColor)            '背景色を設定する
    Ret = Api_TextOut(hDCE, 0, 0, txt, cbStr)       '指定位置にtxt表示

    x = 20                                          'Picture1内のx位置
    y = 16                                          'Picture1内のy位置
    txt = "PICTURE1"
    cbStr = Len(txt)

    rct.Top = 0
    rct.Left = 0
    rct.Right = Picture1.GetWidth - 2               '左右ライン分減算
    rct.Bottom = Picture1.GetHeight - 2             '上下ライン分減算

    Ret = Api_FillRect(hDCP, rct, colBrush)         '長方形を設定
    Ret = Api_SetBkMode(hDCP, OPAQUE)               'Picture1の背景色のモードを設定する
    Ret = Api_SetBkColor(hDCP, rgbColor)            '背景色を設定する
    Ret = Api_TextOut(hDCP, x, y, txt, cbStr)       '指定位置にtxt表示

    rct.Top = 2
    rct.Left = 2
    rct.Right = Combo1.GetWidth - 22                '3D枠+▼分減算
    rct.Bottom = Combo1.GetHeight - 2                   '〃

    txt = "COMBO1"
    cbStr = Len(txt)
    Ret = Api_FillRect(hDCC, rct, colBrush)         '長方形を設定
    Ret = Api_SetBkMode(hDCC, OPAQUE)               'Combo1の背景色のモードを設定する
    Ret = Api_SetBkColor(hDCC, rgbColor)            '背景色を設定する
    Ret = Api_TextOut(hDCC, 2, 2, txt, cbStr)       '指定位置にtxt表示

    rct.Top = 0
    rct.Left = 0
    rct.Right = List1.GetWidth - 6                  '3D枠分減算
    rct.Bottom = List1.GetHeight - 6                '〃
    txt = "LIST1"
    cbStr = Len(txt)
    Ret = Api_FillRect(hDCL, rct, colBrush)         '長方形を設定
    Ret = Api_SetBkMode(hDCL, OPAQUE)               'List1の背景色のモードを設定する
    Ret = Api_SetBkColor(hDCL, rgbColor)            '背景色を設定する
    Ret = Api_TextOut(hDCL, 0, 0, txt, cbStr)       '指定位置にtxt表示

    rct.Top = 0
    rct.Left = 16
    rct.Right = Check1.GetWidth
    rct.Bottom = Check1.GetHeight

    txt = "CHECK1"
    cbStr = Len(txt)
    Ret = Api_FillRect(hDCK, rct, colBrush)         '長方形を設定
```

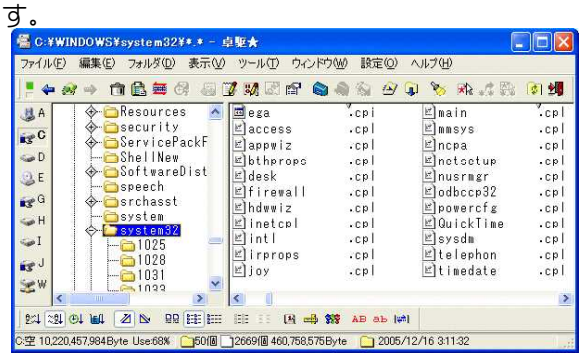
```
    Ret = Api_SetBkMode(hDCK, OPAQUE)              'Check1の背景色のモードを設定する
    Ret = Api_SetBkColor(hDCK, rgbColor)           '背景色を設定する
    Ret = Api_TextOut(hDCK, 18, 2, txt, cbStr)     '指定位置にtxt表示
End Sub

'=====================================
'=
'=====================================
Declare Sub Scroll_Change edecl ()
Sub Scroll_Change()
    r = HScroll1.GetScrollPos
    g = HScroll2.GetScrollPos
    b = HScroll3.GetScrollPos
    rgbColor = RGB(r, g, b)

    Text(3).SetWindowText Trim$(Str$(r))
    Text(4).SetWindowText Trim$(Str$(g))
    Text(5).SetWindowText Trim$(Str$(b))
    txt1$ = "RGB( " & Trim$(Str$(r)) & ", " & Trim$(Str$(g)) & ", " & Trim$(Str$(b)) & ")"
    txt2$ = "RGB( &&H" & Hex$(r) & ", &&H" & Hex$(g) & ", &&H" & Hex$(b) & ")"
    Text(7).SetWindowText txt1$ & Chr$(13,10) & txt2$
    BackColorSet
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    HScroll1.SetScrollPos 128
    HScroll2.SetScrollPos 128
    HScroll3.SetScrollPos 128

    hDCT = Api_GetDC(Text(6).GethWnd)              'Text(6)のDC取得
    hDCE = Api_GetDC(Edit1.GethWnd)                'Edit1のDC取得
    hDCP = Api_GetDC(Picture1.GethWnd)             'Picture1のDC取得
    hDCC = Api_GetDC(Combo1.GethWnd)               'Combo1のDC取得
    hDCL = Api_GetDC(List1.GethWnd)                'List1のDC取得
    hDCK = Api_GetDC(Check1.GethWnd)

    rgbColor = RGB(HScroll1.GetScrollPos, HScroll2.GetScrollPos, HScroll3.GetScrollPos)
    BackColorSet
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_QueryClose edecl (Cancel%, Mode%)
Sub MainForm_QueryClose(Cancel%, Mode%)
    Var Ret As Long

    If Cancel% = 0 Then
        Ret = Api_ReleaseDC(Text(6).GethWnd, hDCT)
        Ret = Api_ReleaseDC(Edit1.GethWnd, hDCE)
        Ret = Api_ReleaseDC(Picture1.GethWnd, hDCP)
        Ret = Api_ReleaseDC(Combo1.GethWnd, hDCC)
        Ret = Api_ReleaseDC(List1.GethWnd, hDCL)
        End
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

コントロールパネルのアイテムを呼び出す(Ⅰ)

コントロールパネルのアイテムを直接呼び出します。
WinExec 指定されたアプリケーションを実行

cplファイルを実行します。osにより無い場合もあります。
cplファイルは、System32のフォルダに保存されています。





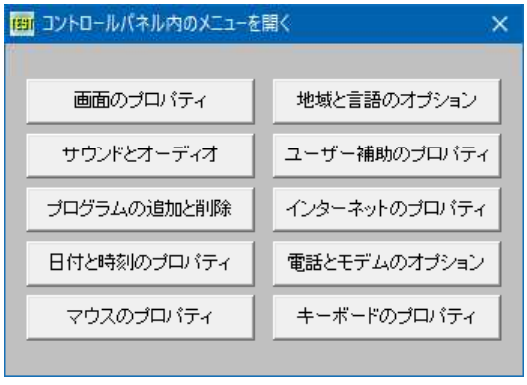

```
'=======================================
'= コントロールパネルアイテムを呼び出す
'=    (WinExec.bas)
'=======================================
#include "Windows.bi"

' 指定されたアプリケーションを実行
Declare Function Api_WinExec& Lib "kernel32" Alias "WinExec" (ByVal lpCmdLine$, ByVal
nCmdShow&)

Var Shared Radio(18) As Object
Var Shared Button1 As Object

For i = 0 To 18
    Radio(i).Attach GetDlgItem("Radio" & Trim$(Str$(i + 1))) : Radio(i).SetFontSize 12
Next i
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

ShowWindow -1

'=======================================
'=
'=======================================
Declare Function Index bdecl () As Integer
Function Index()
    Index = Val(Mid$(GetDlgRadioSelect("Radio1"), 6)) - 1
End Function

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var CmdLine As String
    Var Ret As Long

    Select Case Index
        Case 0    'コントロールパネル
            CmdLine = "control"
        Case 1    'アプリケーションの追加と削除のプロパティ(フォルダオプション:Appwiz.cpl @1)
            CmdLine = "Rundll32.exe Shell32.dll, Control_RunDLL Appwiz.cpl"
        Case 2    '画面のプロパティ
            CmdLine = "Rundll32.exe Shell32.dll, Control_RunDLL Desk.cpl"
        Case 3    'インターネットのプロパティ
            CmdLine = "Rundll32.exe Shell32.dll, Control_RunDLL Inetcpl.cpl"
        Case 4    '地域の言語のオプション
            CmdLine = "Rundll32.exe Shell32.dll, Control_RunDLL Intl.cpl"
        Case 5    'ゲームコントローラー
            CmdLine = "Rundll32.exe Shell32.dll, Control_RunDLL Joy.cpl"
```

```
        Case 6    'マウスのプロパティ(キーボード:Main.cpl @1)
            CmdLine = "Rundll32.exe Shell32.dll, Control_RunDLL Main.cpl"
        Case 7    'メールとファックス
            CmdLine = "Rundll32.exe Shell32.dll, Control_RunDLL Mlcfg32.cpl"
        Case 8    'サウンドとオーディオデバイスのプロパティ
            CmdLine = "Rundll32.exe Shell32.dll, Control_RunDLL Mmsys.cpl"
        Case 9    '電話とモデムのオプション
            CmdLine = "Rundll32.exe Shell32.dll, Control_RunDLL Modem.cpl"
        Case 10    '赤外線も似た
            CmdLine = "Rundll32.exe Shell32.dll, Control_RunDLL Infrared.cpl"
        Case 11    'ネットワーク接続(ネットワークセットアップウィザード:NetSetup.cpl)
            CmdLine = "Rundll32.exe Shell32.dll, Control_RunDLL Ncpa.cpl"
        Case 12    'パスワードのプロパティ
            CmdLine = "Rundll32.exe Shell32.dll, Control_RunDLL Password.cpl"
        Case 13    '伝下の婦ションのプロパティ
            CmdLine = "Rundll32.exe Shell32.dll, Control_RunDLL Powercfg.cpl"
        Case 14    'スキャナとカメラのプロパティ
            CmdLine = "Rundll32.exe Shell32.dll, Control_RunDLL Sticpl.cpl"
        Case 15    'システムのプロパティ
            CmdLine = "Rundll32.exe Shell32.dll, Control_RunDLL Sysdm.cpl"
        Case 16    '日付と時刻のプロパティ
            CmdLine = "Rundll32.exe Shell32.dll, Control_RunDLL Timedate.cpl"
        Case 17    'Microsoftワークグループポスト管理
            CmdLine = "Rundll32.exe Shell32.dll, Control_RunDLL Wgpocpl.cpl"
        Case 18    'ユーザー補助のオプション
            CmdLine = "Rundll32.exe Shell32.dll, Control_RunDLL Access.cpl"
    End Select

    Ret = Api_WinExec(CmdLine, 0)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

コントロールパネルのアイテムを呼び出す(II)

コントロールパネル内の各アイテムはCPLファイル形式でWindowsのシステムフォルダに保存されています。
それらを直接呼び出してみます。

CPlApplet_Desk 画面のプロパティ
CPlApplet_Intl 地域と言語のオプション
CPlApplet_MMSys サウンドとオーディオデバイスのプロパティ
CPlApplet_Access ユーザー補助のプロパティ
CPlApplet_AppWiz プログラムの追加と削除
CPlApplet_InetCpl インターネットのプロパティ
CPlApplet_Telephon 電話とモデムのオプション
CPlApplet_TimeDate 日付と時刻のプロパティ
CPlApplet_Main マウスのプロパティ・キーボードのプロパティ



```
'====================================
'= コントロールパネル内のメニューを開く
'=    (CPlApplet.bas)
'====================================
#include "Windows.bi"
Type CPLINFO
    idIcon     As Long
    idName     As Long
    idInfo     As Long
    lData      As Long
End Type

' 画面のプロパティ
Declare Function CPlApplet_Desk& Lib "desk.cpl" Alias "CPlApplet" (ByVal GethWndCPl&,
ByVal uMsg&, ByVal lParam1&, ByVal lParam2&)

' 地域と言語のオプション
Declare Function CPlApplet_Intl& Lib "intl.cpl" Alias "CPlApplet" (ByVal GethWndCPl&,
ByVal uMsg&, ByVal lParam1&, ByVal lParam2&)

' サウンドとオーディオデバイスのプロパティ
Declare Function CPlApplet_MMSys& Lib "mmsys.cpl" Alias "CPlApplet" (ByVal GethWndCPl&,
ByVal uMsg&, ByVal lParam1&, ByVal lParam2&)

' ユーザー補助のプロパティ
Declare Function CPlApplet_Access& Lib "access.cpl" Alias "CPlApplet" (ByVal
GethWndCPl&, ByVal uMsg&, ByVal lParam1&, ByVal lParam2&)

' プログラムの追加と削除
Declare Function CPlApplet_AppWiz& Lib "appwiz.cpl" Alias "CPlApplet" (ByVal
GethWndCPl&, ByVal uMsg&, ByVal lParam1&, ByVal lParam2&)

' インターネットのプロパティ
Declare Function CPlApplet_InetCpl& Lib "inetcpl.cpl" Alias "CPlApplet" (ByVal
GethWndCPl&, ByVal uMsg&, ByVal lParam1&, ByVal lParam2&)

' 電話とモデムのオプション
Declare Function CPlApplet_Telephon& Lib "telephon.cpl" Alias "CPlApplet" (ByVal
GethWndCPl&, ByVal uMsg&, ByVal lParam1&, ByVal lParam2&)

' 日付と時刻のプロパティ
Declare Function CPlApplet_TimeDate& Lib "timedate.cpl" Alias "CPlApplet" (ByVal
GethWndCPl&, ByVal uMsg&, ByVal lParam1&, ByVal lParam2&)

' マウスのプロパティ・キーボードのプロパティ
Declare Function CPlApplet_Main& Lib "main.cpl" Alias "CPlApplet" (ByVal GethWndCPl&,
ByVal uMsg&, ByVal lParam1&, ByVal lParam2&)

#define CPL_INIT 1
#define CPL_GETCOUNT 2
#define CPL_INQUIRE 3
#define CPL_SELECT 4
#define CPL_DBLCLK 5
#define CPL_STOP 6
#define CPL_EXIT 7
#define CPL_NEWINQUIRE 8

Var Shared ci As CPLINFO

Var Shared Button(9) As Object

For i = 0 To 9
    Button(i).Attach GetDlgItem("Button" & Trim$(Str$(i + 1)))
    Button(i).SetFontSize 12
Next

'====================================
'=
'====================================
Declare Sub MainCplApplet(lParam1 As Long)
```

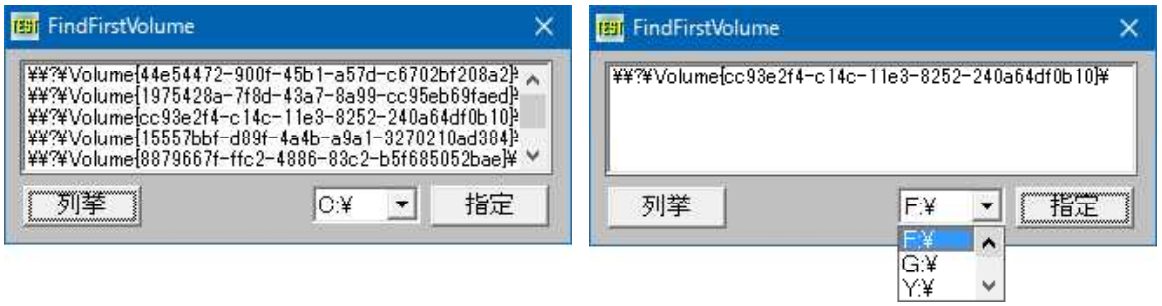
```
Sub MainCplApplet(lParam1 As Long)
    Var Ret As Long

    If CPlApplet_Main(GethWnd, CPL_INIT, 0, 0) <> 0 Then
        Ret = CPlApplet_Main(GethWnd, CPL_INQUIRE, lParam1, VarAdr(ci))
        Ret = CPlApplet_Main(GethWnd, CPL_DBLCLK, lParam1, ci.lData)
        Ret = CPlApplet_Main(GethWnd, CPL_STOP, 0, ci.lData)
        Ret = CPlApplet_Main(GethWnd, CPL_EXIT, 0, 0)
    End If
End Sub

'=====================================
'= 画面のプロパティ
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    If CPlApplet_Desk(GethWnd, CPL_INIT, 0, 0) <> 0 Then
        Ret = CPlApplet_Desk(GethWnd, CPL_INQUIRE, 0, VarAdr(ci))
        Ret = CPlApplet_Desk(GethWnd, CPL_DBLCLK, 0, ci.lData)
        Ret = CPlApplet_Desk(GethWnd, CPL_STOP, 0, ci.lData)
        Ret = CPlApplet_Desk(GethWnd, CPL_EXIT, 0, 0)
    End If
End Sub

'=====================================
'= サウンドとオーディオデバイスのプロパティ
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Ret As Long

    If CPlApplet_MMSys(GethWnd, CPL_INIT, 0, 0) <> 0 Then
        Ret = CPlApplet_MMSys(GethWnd, CPL_INQUIRE, 0, VarAdr(ci))
        Ret = CPlApplet_MMSys(GethWnd, CPL_DBLCLK, 0, ci.lData)
        Ret = CPlApplet_MMSys(GethWnd, CPL_STOP, 0, ci.lData)
        Ret = CPlApplet_MMSys(GethWnd, CPL_EXIT, 0, 0)
    End If
End Sub

'=====================================
'= プログラムの追加と削除
'=====================================
Declare Sub Button3_on edecl ()
Sub Button3_on()
    Var Ret As Long

    If CPlApplet_AppWiz(GethWnd, CPL_INIT, 0, 0) <> 0 Then
        Ret = CPlApplet_AppWiz(GethWnd, CPL_INQUIRE, 0, VarAdr(ci))
        Ret = CPlApplet_AppWiz(GethWnd, CPL_DBLCLK, 0, ci.lData)
        Ret = CPlApplet_AppWiz(GethWnd, CPL_STOP, 0, ci.lData)
        Ret = CPlApplet_AppWiz(GethWnd, CPL_EXIT, 0, 0)
    End If
End Sub

'=====================================
'= 日付と時刻のプロパティ
'=====================================
Declare Sub Button4_on edecl ()
Sub Button4_on()
    Var Ret As Long

    If CPlApplet_TimeDate(GethWnd, CPL_INIT, 0, 0) <> 0 Then
        Ret = CPlApplet_TimeDate(GethWnd, CPL_INQUIRE, 0, VarAdr(ci))
        Ret = CPlApplet_TimeDate(GethWnd, CPL_DBLCLK, 0, ci.lData)
        Ret = CPlApplet_TimeDate(GethWnd, CPL_STOP, 0, ci.lData)
        Ret = CPlApplet_TimeDate(GethWnd, CPL_EXIT, 0, 0)
    End If
End Sub
```

```
'=====================================
'= マウスのプロパティ
'=====================================
Declare Sub Button5_on edecl ()
Sub Button5_on()

    MainCplApplet(0)
End Sub

'=====================================
'= 地域と言語のオプション
'=====================================
Declare Sub Button6_on edecl ()
Sub Button6_on()
    Var Ret As Long

    If CPlApplet_Intl(GethWnd, CPL_INIT, 0, 0) <> 0 Then
        Ret = CPlApplet_Intl(GethWnd, CPL_INQUIRE, 0, VarAdr(ci))
        Ret = CPlApplet_Intl(GethWnd, CPL_DBLCLK, 0, ci.lData)
        Ret = CPlApplet_Intl(GethWnd, CPL_STOP, 0, ci.lData)
        Ret = CPlApplet_Intl(GethWnd, CPL_EXIT, 0, 0)
    End If
End Sub

'=====================================
'= ユーザー補助のプロパティ
'=====================================
Declare Sub Button7_on edecl ()
Sub Button7_on()
    Var Ret As Long

    If CPlApplet_Access(GethWnd, CPL_INIT, 0, 0) <> 0 Then
        Ret = CPlApplet_Access(GethWnd, CPL_INQUIRE, 0, VarAdr(ci))
        Ret = CPlApplet_Access(GethWnd, CPL_DBLCLK, 0, ci.lData)
        Ret = CPlApplet_Access(GethWnd, CPL_STOP, 0, ci.lData)
        Ret = CPlApplet_Access(GethWnd, CPL_EXIT, 0, 0)
    End If
End Sub

'=====================================
'= インターネットのプロパティ
'=====================================
Declare Sub Button8_on edecl ()
Sub Button8_on()
    Var Ret As Long

    If CPlApplet_InetCpl(GethWnd, CPL_INIT, 0, 0) <> 0 Then
        Ret = CPlApplet_InetCpl(GethWnd, CPL_INQUIRE, 0, VarAdr(ci))
        Ret = CPlApplet_InetCpl(GethWnd, CPL_DBLCLK, 0, ci.lData)
        Ret = CPlApplet_InetCpl(GethWnd, CPL_STOP, 0, ci.lData)
        Ret = CPlApplet_InetCpl(GethWnd, CPL_EXIT, 0, 0)
    End If
End Sub

'=====================================
'= 電話とモデムのオプション
'=====================================
Declare Sub Button9_on edecl ()
Sub Button9_on()
    Var Ret As Long

    If CPlApplet_Telephon(GethWnd, CPL_INIT, 0, 0) <> 0 Then
        Ret = CPlApplet_Telephon(GethWnd, CPL_INQUIRE, 0, VarAdr(ci))
        Ret = CPlApplet_Telephon(GethWnd, CPL_DBLCLK, 0, ci.lData)
        Ret = CPlApplet_Telephon(GethWnd, CPL_STOP, 0, ci.lData)
        Ret = CPlApplet_Telephon(GethWnd, CPL_EXIT, 0, 0)
    End If
End Sub
```

```
'=====================================
'= キーボードのプロパティ
'=====================================
Declare Sub Button10_on edecl ()
Sub Button10_on()
    MainCplApplet(1)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

コンピュータ上のボリューム名を取得

GetLogicalDrives 利用可能ディスクドライブ取得
FindVolumeClose 指定したボリューム検索ハンドルを閉じる
FindFirstVolume コンピュータ上のボリューム名を返す
FindNextVolume FindFirstVolume 関数により開始したボリューム検索を継続
lstrlenW (Null文字で終了する)UNICODE文字列の文字数を返す
GetVolumeNameForVolumeMountPoint マウントポイントまたはルートディレクトリを取得し、それに対応する一意のボリューム名を返す




```
'=====================================
'= コンピュータ上のボリューム名を取得
'=    (FindFirstVolume.bas)
'=====================================
#include "Windows.bi"

' 利用可能ディスクドライブ取得
Declare Function Api_GetLogicalDrives& Lib "kernel32" Alias "GetLogicalDrives" ()

' 指定したボリューム検索ハンドルを閉じる
Declare Function Api_FindVolumeClose& Lib "kernel32" Alias "FindVolumeClose" (ByVal
hFindVolume&)

' コンピュータ上のボリューム名を返す
Declare Function Api_FindFirstVolume& Lib "kernel32" Alias "FindFirstVolumeA" (ByVal
lpszVolumeName$, ByVal cchBufferLength&)

' FindFirstVolume 関数により開始したボリューム検索を継続
Declare Function Api_FindNextVolume& Lib "kernel32" Alias "FindNextVolumeA" (ByVal
hFindVolume&, ByVal lpszVolumeName$, ByVal cchBufferLength&)

' (Null文字で終了する)UNICODE文字列の文字数を返す
Declare Function Api_lstrlenW& Lib "kernel32" Alias "lstrlenW" (ByVal lpString&)

' マウントポイントまたはルートディレクトリを取得し、それに対応する一意のボリューム名を返す
Declare Function Api_GetVolumeNameForVolumeMountPoint& Lib "kernel32" Alias
"GetVolumeNameForVolumeMountPointA" (ByVal lpszVolumeMountPoint$, ByVal
lpszVolumeName$, ByVal cchBufferLength&)
```

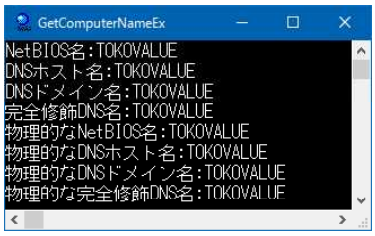
```
Var Shared List1 As Object
Var Shared Combo1 As Object
Var Shared Button1 As Object
Var Shared Button2 As Object
List1.Attach GetDlgItem("List1") : List1.SetFontSize 12 : List1.SetWindowSize 306, 66
Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14

'=====================================
'=
'=====================================
Declare Function TrimNull(startstr As String) As String
Function TrimNull(startstr As String) As String
    TrimNull = Left$(startstr, Api_lstrlenW(StrAdr(startstr)))
End Function

'=====================================
'=
'=====================================
Declare Function GetVolumeFromDrive(sVolumeMountPoint As String) As String
Function GetVolumeFromDrive(sVolumeMountPoint As String) As String
    Var buff As String
    Var cbbuff As Long

    buff = Space$(1024)
    cbbuff = Len(buff)

    If Api_GetVolumeNameForVolumeMountPoint(sVolumeMountPoint, buff, cbbuff) <> 0 Then
        GetVolumeFromDrive = TrimNull(buff)
    End If
End Function

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var Drive As String
    Var i As Integer

    Drives = Api_GetLogicalDrives()
    If Drives = 0 Then Exit Sub

    For i = 0 To 25
        If (Drives And 1) = 1 Then
            Drive = Chr$(65 + i)
            Drive = Drive & ":¥"
            Combo1.AddString Drive
        End If
        Drives = Drives ¥ 2
    Next i
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var buff As String
    Var hVolume As Long
    Var cbbuff As Long
    Var sVolumeMountPoint As String
    Var Ret As Long

    buff = Space$(4096)
    cbbuff = Len(buff)

    hVolume = Api_FindFirstVolume(buff, cbbuff)
```

```
    List1.ResetContent

    Do
        sVolumeMountPoint = TrimNull(buff)
        List1.AddString sVolumeMountPoint
        buff = Space$(4096)
        cbbuff = Len(buff)
    Loop While Api_FindNextVolume(hVolume, buff, cbbuff)

    List1.AddString ""

    Ret = Api_FindVolumeClose(hVolume)
End Sub

'=======================================
'=
'=======================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    List1.ResetContent
    List1.AddString GetVolumeFromDrive(Left$(Combo1.GetText(Combo1.GetCursel), 3))
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

コンピュータ名取得(Ⅰ)

コンピュータ名を取得します。GetComputerNameとの相違点は、取得する名前の種類を指定することができることです。

GetComputerNameEx コンピュータ名を取得



ComputerNameNetBIOS	NetBIOS名を取得します。ローカルコンピュータがクラスタ内のノードの1つである場合は、クラスタのNetBIOS名を取得します。
ComputerNameDnsHostname	ローカルコンピュータのDNSホスト名を取得します。ローカルコンピュータがクラスタ内のノードの1つである場合は、クラスタのDNSホスト名を取得します。
ComputerNameDnsDomain	ローカルコンピュータに割り当てられている DNS ドメインの名前を取得します。ローカルコンピュータがクラスタ内のノードの1つである場合は、クラスタのDNSドメイン名を取得します。
ComputerNameDnsFullyQualified	ローカルコンピュータを一意に識別する完全修飾 DNS 名を取得します。この名前は、DNSホスト名とDNSドメイン名をHostName.DomainNameの形式で組み合わせたものです。ローカルコンピュータがクラスタ内のノードの1つである場合は、クラスタの完全修飾DNS名を取得します。
ComputerNamePhysicalNetBIOS	ローカルコンピュータのNetBIOS名を取得します。ローカルコンピュータがクラスタ内のノードの1つである場合は、クラスタではなく、ローカルコンピュータのNetBIOS名を取得します。
ComputerNamePhysicalDnsHostname	ローカルコンピュータのDNSホスト名を取得します。ローカルコンピュータがクラスタ内のノードの1つである場合は、クラスタではなく、ローカルコンピュータのDNSホスト名を取得します。
ComputerNamePhysicalDnsDomain	ローカルコンピュータに割り当てられているDNSドメインの名前を取得します。ローカルコンピュータがクラスタ内のノードの1つである場合は、クラスタではなく、ローカルコンピュータのDNSドメイン名を取得します。

ComputerNamePhysicalDnsFullyQualified	コンピュータを一意に識別する完全修飾DNS名を取得します。ローカルコンピュータがクラスタ内のノードの1つである場合は、クラスタではなく、ローカルコンピュータの完全修飾DNS名を取得します。

```
'=====================================
'= コンピュータ名を取得
'=    (GetComputerNameEx.bas)
'=====================================

' コンピュータ名を取得
Declare Function Api_GetComputerNameEx& Lib "kernel32" Alias "GetComputerNameExA" (ByVal
NameType&, ByVal lpBuffer$, lpnSize&)

'=====================================
'= Null[Chr$(0)]を取り除く
'=====================================
Declare Function TrimNull (item As String) As String
Function TrimNull(item As String) As String
    Var ePos As Integer

    ePos = instr(item, Chr$(0))
    If ePos Then
        TrimNull = left$(item, ePos - 1)
    Else
        TrimNull = item
    End If
End Function

'------------------------------
Var ComputerName As String * 128
Var txt As String
Var Ret As Long

Ret = Api_GetComputerNameEx(ComputerNameNetBIOS, ComputerName, len(ComputerName))
txt = "NetBIOS名" : GoSub *dsp

Ret = Api_GetComputerNameEx(ComputerNameDnsHostname, ComputerName, len(ComputerName))
txt = "DNSホスト名" : GoSub *dsp

Ret = Api_GetComputerNameEx(ComputerNameDnsDomain, ComputerName, len(ComputerName))
txt = "DNSドメイン名" : GoSub *dsp

Ret = Api_GetComputerNameEx(ComputerNameDnsFullyQualified, ComputerName,
len(ComputerName))
txt = "完全修飾DNS名" : GoSub *dsp

Ret = Api_GetComputerNameEx(ComputerNamePhysicalNetBIOS, ComputerName,
len(ComputerName))
txt = "物理的なNetBIOS名" : GoSub *dsp

Ret = Api_GetComputerNameEx(ComputerNamePhysicalDnsHostname, ComputerName,
len(ComputerName))
txt = "物理的なDNSホスト名" : GoSub *dsp

Ret = Api_GetComputerNameEx(ComputerNamePhysicalDnsDomain, ComputerName,
len(ComputerName))
txt = "物理的なDNSドメイン名" : GoSub *dsp

Ret = Api_GetComputerNameEx(ComputerNamePhysicalDnsFullyQualified, ComputerName,
len(ComputerName))
txt = "物理的な完全修飾DNS名" : GoSub *dsp

Stop
End

*dsp
    If Ret <> 0 Then
        Print txt & ":" & TrimNull(ComputerName)
    End If
return
```

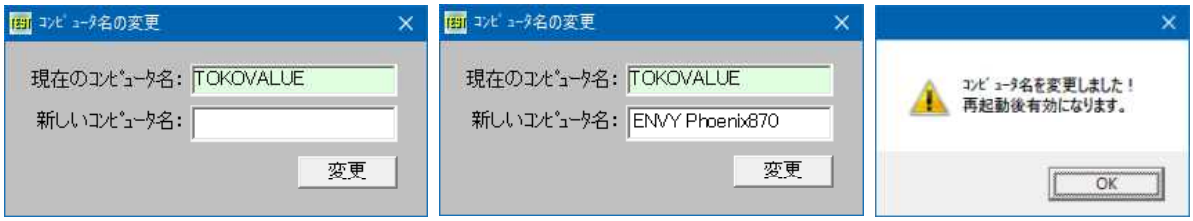
コンピュータ名取得（II）

コンピュータ名を取得します。
Environ$ 環境変数を獲得


```
'=====================================
'= コンピュータ名取得
'=====================================
Declare Function GetComputerName() As String
Function GetComputerName() As String
    GetComputerName = Environ$("ComputerName")
End Function

Print GetComputerName()

Stop
End
```

コンピュータ名の変更

コンピュータ名を変更します。
GetComputerName コンピュータ名取得
SetComputerName コンピュータ名設定



起動すると現在のコンピュータ名が表示されます。新しいコンピュータ名を入力し変更ボタンをクリックすると再起動後有効になります。

```
'=====================================
'=  コンピュータ名の変更
'=    (SetComputerName.bas)
'=====================================
#include "Windows.bi"

' コンピュータの名前を文字列として取得
Declare Function Api_GetComputerName& Lib "kernel32" Alias "GetComputerNameA" (ByVal
lpBuffer$, nSize&)

' コンピュータ名を変更
Declare Function Api_SetComputerName& Lib "kernel32" Alias "SetComputerNameA" (ByVal
lpComputerName$)

Var Shared Text(2) As Object
Var Shared Edit1 As Object
Var Shared Button1 As Object

For i = 0 To 2
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i+1)))
    Text(i).SetFontSize 14
Next
```

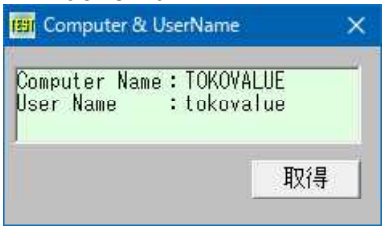
```
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var pcName$ As String
    Var Leng As Long
    Var Res As Long

    pcName$ = String$(250, Chr$(0))
    Leng = Len(pcName$)
    Res = Api_GetComputerName(pcName$, Leng)
    pcName$ = Left$(pcName$, InStr(pcName$, Chr$(0)) - 1)

    Text(2).SetWindowText pcName$
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var sNewName$ As String
    Var Res As Long

    sNewName$ = GetDlgItemText("Edit1")
    If sNewName$ = "" Then Exit Sub

    Res = Api_SetComputerName(sNewName$)
    Res = MessageBox("", "ｺﾝﾋﾟｭｰﾀ名を変更しました！" & Chr$(13,10) & "再起動後有効になります。",
0, 2)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

コンピュータ名･ユーザ名の取得

コンピュータ名およびユーザ名を取得します。
GetComputerName コンピュータ名取得
GetUserName ユーザ名を取得

Windows XP

Windows 10

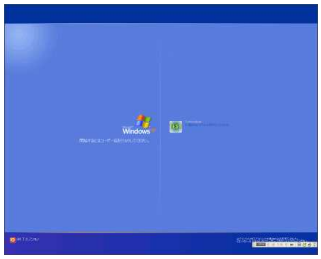



```
'=====================================
'=  コンピュータ名･ユーザー名取得
'=    (ComUsrName.bas)
'=====================================
#include "Windows.bi"
```

```
' コンピュータの名前を文字列として取得
Declare Function Api_GetComputerName& Lib "Kernel32" Alias "GetComputerNameA" (ByVal
lpBuffer$, nSize&)

' ユーザー名を取得
Declare Function Api_GetUserName& Lib "advapi32" Alias "GetUserNameA" (ByVal lpBuffer$,
nSize&)

Var Shared Text1 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var PcName As String
    Var UserName As String
    Var Length As Long
    Var txt As String
    Var Ret As Long

    PcName = String$(250, Chr$(0))
    Length = Len(PcName)
    Ret = Api_GetComputerName(PcName, Length)
    PcName = Left$(PcName, InStr(PcName, Chr$(0)) - 1)

    UserName = String$(250, Chr$(0))
    Length = Len(UserName)
    Ret = Api_GetUserName(UserName, Length)
    UserName = Left$(UserName, InStr(UserName, Chr$(0)) - 1)

    txt = "Computer Name:" & PcName & Chr$(13, 10) & "User Name    :" & UserName
    Text1.SetWindowtext txt
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

コンピュータをロックする

LockWorkStation コンピュータをロックします。



Windows2000/WindowsXp

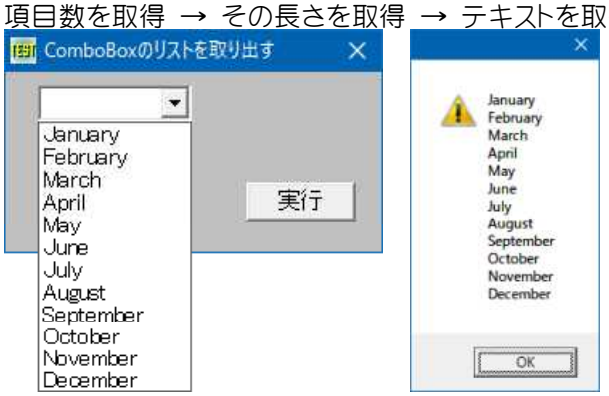
```
'=====================================
'= コンピュータをロックする
'=    (LockWorkStation.bas)
'=====================================
```

```
' コンピュータをロック
Declare Function Api_LockWorkStation& Lib "user32" Alias "LockWorkStation" ()

Var Ret As Long

Ret = Api_LockWorkStation

Stop
End
```

コンボボックス内のリストを取り出す

コンボボックス内のリストを取り出します。
SendMessage 指定のウィンドウにメッセージを送る
CB_GETCOUNT(&H146) 項コンボボックスのリストボックスの項目数を取得する
CB_GETLBTEXT(&H148) コンボボックスのリストボックスから文字列を取得する
CB_GETLBTEXTLEN(&H149) コンボボックスのリストボックス文字列の長さを取得する

項目数を取得 → その長さを取得 → テキストを取得



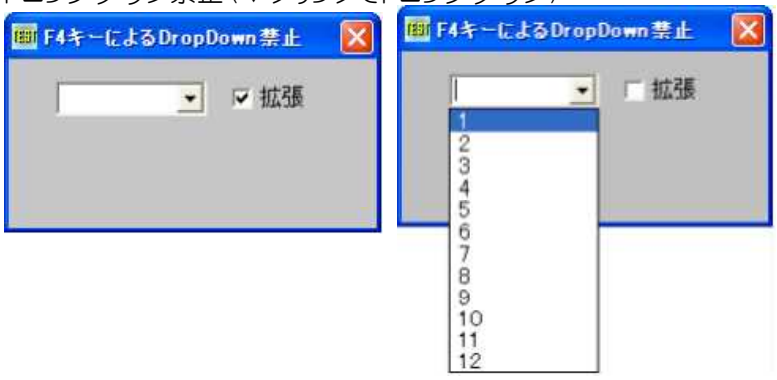

```
'=====================================
'= コンボボックスのリストを取り出す
'=    (SendMessage4.bas)
'=====================================
#include "Windows.bi"

' ウィンドウにメッセージを送信
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

#define CB_GETCOUNT &H146                    'コンボボックスのリストボックスの項目数を取得する
#define CB_GETLBTEXT &H148                   'コンボボックスのリストボックスから文字列を取得する
#define CB_GETLBTEXTLEN &H149                'コンボボックスのリストボックス文字列の長さを取得する

#define vbCrLf (Chr$(13) & Chr$(10))         'キャリッジリターンとラインフィード(¥r¥n)

Var Shared Combo1 As Object
Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Combo1.AddString "January"
    Combo1.AddString "February"
    Combo1.AddString "March"
    Combo1.AddString "April"
    Combo1.AddString "May"
    Combo1.AddString "June"
    Combo1.AddString "July"
    Combo1.AddString "August"
    Combo1.AddString "September"
    Combo1.AddString "October"
```

```
    Combo1.AddString "November"
    Combo1.AddString "December"
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var num As Long
    Var i As Integer
    Var txt As String
    Var Buffer As String
    Var length As Long

    'コンボボックスの項目数を取得
    num = Api_SendMessage(Combo1.GethWnd, CB_GETCOUNT, 0, 0)

    'リストを調べる
    For i = 0 To num - 1

        '項目の長さを取得
        length = Api_SendMessage(Combo1.GethWnd, CB_GETLBTEXTLEN, i, 0)

        'バッファの確保
        Buffer = Space$(length + 1)

        'テキストを取得
        length = Api_SendMessage(Combo1.GethWnd, CB_GETLBTEXT, i, Buffer)
        txt = txt & Left$(Buffer, length) & vbCrLf
    Next i

    A% = MessageBox("", txt, 0, 2)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

コンボボックスのF4キーによるDropDown禁止

通常、コンボボックスはF4キーでドロップダウンしますが、それを禁止します。
SendMessage ウィンドウにメッセージを送信
CB_SETEXTENDEDUI デフォルトまたは拡張のユーザーインターフェイスを設定

ドロップダウン禁止（▼クリックでドロップダウン）



F4キーでドロップダウン

```
'=====================================
'= コンボボックスのF4キーによるドロップダウン禁止
'=====================================
#include "Windows.bi"

' ウィンドウにメッセージを送信。
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, ByVal lParam&)

#define CB_GETEXTENDEDUI &H156              'コンボボックスが拡張インターフェイスを持つかを判断する
#define CB_SETEXTENDEDUI &H155              'デフォルトまたは拡張のユーザーインターフェイスを設定

Var Shared Combo1 As Object
Var Shared Check1 As Object

Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
Check1.Attach GetDlgItem("Check1")

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    For i% = 1 To 12
        Combo1.AddString Str$(i%)
    Next
    Combo1.SetFocus
End Sub

'=====================================
'=
'=====================================
Declare Sub Check1_on edecl ()
Sub Check1_on()
    Var Ret As Long

    If Check1.GetCheck Then
        Ret = Api_SendMessage(Combo1.GethWnd, CB_SETEXTENDEDUI, 1, 0)
    Else
        Ret = Api_SendMessage(Combo1.GethWnd, CB_SETEXTENDEDUI, 0, 0)
    End If
    Combo1.SetFocus
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

コンボボックスのエディットウィンドウハンドルを取得

FindWindowEx クラス名 、または キャプションを与えてウィンドウのハンドルを取得
SendMessage ウィンドウにメッセージを送信
SetWindowText ウィンドウのタイトルを変更

```
'=====================================
'= コンボボックスのエディットウィンドウハンドルを取得
'=   (FindWindowEx2.bas)
'=====================================
#include "Windows.bi"

#define CB_ADDSTRING &H143                    'コンボボックスのリストボックスに文字列を追加する
#define CB_SETITEMDATA &H151                  'アプリケーション定義の値を設定する

' クラス名 、または キャプションを与えてウィンドウのハンドルを取得
Declare Function Api_FindWindowEx& Lib "user32" Alias "FindWindowExA" (ByVal
hWndParent&, ByVal hWndChildAfter&, ByVal lpszClass$, ByVal lpszWindow$)

' ウィンドウにメッセージを送信
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

' ウィンドウのタイトルを変更
Declare Function Api_SetWindowText& Lib "user32" Alias "SetWindowTextA" (ByVal hWnd&,
ByVal lpString$)

Var Shared Combo1 As Object
Var Shared Button1 As Object

Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 12
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hwndEdit As Long
    Var Ret As Long

    hwndEdit = Api_FindWindowEx(Combo1.GethWnd, 0&, ByVal 0, ByVal 0)

    Ret = Api_SetWindowText(hwndEdit, "FindWindowEx 取得！")

    Ret = Api_SendMessage(Combo1.GethWnd, CB_ADDSTRING, 0&, "FindWindowEx:Editハンドル:" &
Str$(hwndEdit))
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

コンボボックスのエディットの選択範囲を指定

SendMessage 指定のウィンドウにメッセージを送る
CB_SETEDITSEL(&H142) コンボボックスのエディットコントロールで選択文字を設定する
CB_SHOWDROPDOWN(&H14F) コンボボックスのリストボックスの表示または非表示を切り替える

```
'=====================================
'= コンボボックスのエディットの選択範囲を指定
'=    (CB_SETEDITSEL.bas)
'=====================================
#include "Windows.bi"

' ウィンドウにメッセージを送信
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

#define CB_SETEDITSEL &H142                 'コンボボックスのエディットコントロールで選択文字を設定
#define CB_SHOWDROPDOWN &H14F               'コンボボックスのリストボックスの表示又は非表示を切替え
Var Shared Edit(1) As Object
Var Shared Text(1) As Object
Var Shared Combo1 As Object
Var Shared Button1 As Object

Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
For i = 0 To 1
    Edit(i).Attach GetDlgItem("Edit" & Trim$(Str$(i + 1))) : Edit(i).SetFontSize 14
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1))) : Text(i).SetFontSize 14
Next
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Function MAKELPARAM(ByVal Start As Byte, ByVal Length As Byte) As Long
Function MAKELPARAM(ByVal Start As Byte, ByVal Length As Byte) As Long

    '下位BYTE値と16ビット左シフトした上位BYTE値の論理和
    MAKELPARAM = CLng(Length) * (2 ^ 16) Or CLng(Start)
End Function

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var mm(11) As String
    Var Ret As Long

    For i = 0 To 11
        Read mm(i)
        Combo1.AddString mm(i)
    Next

    'マウス砂時計解除
    SetMousePointer 0
    Ret = Api_SendMessage(Combo1.GethWnd, CB_SHOWDROPDOWN, 1, ByVal 0)

    Data "January ", "February", "March", "April", "May", "June"
    Data "July", "August", "September", "October", "November", "December"
End Sub
'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Start As Long
    Var Length As Long
    Var SelRange As Long
    Var Ret As Long

    '選択範囲を指定
    Start = Val(Edit(0).GetWindowtext)
    Length = Start + Val(Edit(1).GetWindowText)

    '指定した選択範囲からLPARAMを生成
    SelRange = MAKELPARAM(CByte(Start), CByte(Length))
```

```
    '選択範囲を設定
    Ret = Api_SendMessage(Combo1.GethWnd, CB_SETEDITSEL, 0, ByVal SelRange)
End Sub

'========================================
'=
'========================================
While 1
    WaitEvent
Wend
Stop
End
```

コンボボックスのエディット部高さを設定

SendMessage ウィンドウにメッセージを送信
CB_SETITEMHEIGHT(&H153) コンボボックス内の項目の高さを設定

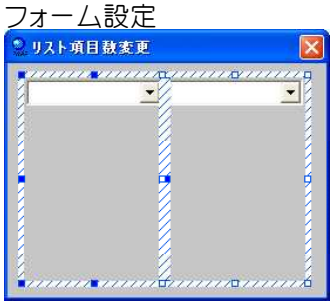

```
'========================================
'= コンボボックスのエディット部高さを設定
'=    (CB_SETITEMHEIGHT.bas)
'========================================
#include "Windows.bi"

' ウィンドウにメッセージを送信
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

#define CB_SETITEMHEIGHT &H153      'コンボボックス内の項目の高さを設定する

Var Shared Text1 As Object
Var Shared Text2 As Object
Var Shared Combo1 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Text2.Attach GetDlgItem("Text2") : Text2.SetFontSize 14
Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'========================================
'=
'========================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    For i = 1 to 10
        Combo1.AddString Str$(i)
    Next
End Sub

'========================================
'=
'========================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var ComboHeight As Long
    Var Item As Long
    Var Ret  As Long
```

```
    '変更する構成要素にエディットボックスを指定
    Item = -1

    'コンボボックスの高さ
    ComboHeight = CLng(Combo1.GetHeight * 1.05)

    '新しいコンボボックスの高さを設定
    Ret = Api_SendMessage(Combo1.GethWnd, CB_SETITEMHEIGHT, Item, ByVal ComboHeight)

    '変更後のエディットボックスの高さを表示
    Text2.SetWindowText Str$(Combo1.GetHeight)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

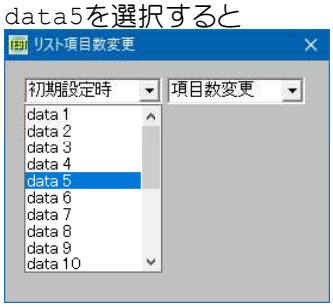
コンボボックスの項目表示数の変更

コンボボックスのリスト項目表示数を変更します。
例ではCOMBO1、2ともに最大10項目表示する高さに設定しています。
起動するとCOMBO1は、最大項目(10)表示で開きます。おなじサイズで設定したCOMBO2は、COMBO1で選択した項目数を表示する高さで開きます。
SendMessage ウィンドウにメッセージを送信
MoveWindow 指定されたウィンドウの位置およびサイズを変更
CB_SHOWDROPDOWN(&H14F) コンボボックスのリスト部、表示・非表示切替
CB_GETITEMHEIGHT(&H154) コンボボックス内の項目の高さを取得

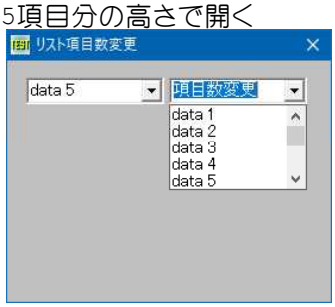
フォーム設定



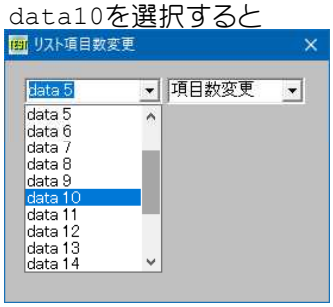
data5を選択すると



5項目分の高さで開く



data10を選択すると



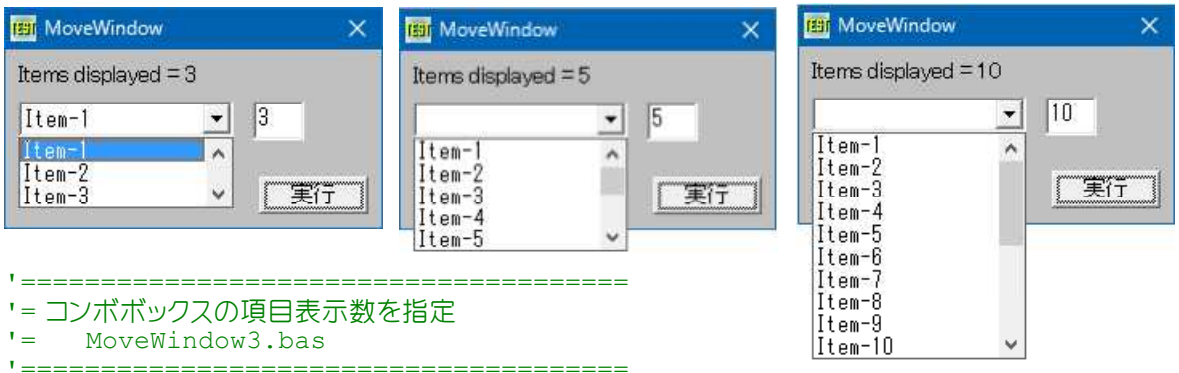
10項目分の高さで開く


```
'=====================================
'= コンボボックスの項目表示数の変更
'=    (ComboItemChange.bas)
'=====================================
#include "Windows.bi"

' ウィンドウにメッセージを送信。この関数は、指定したウィンドウのウィンドウプロシージャが処理を終了するまで制御を返さない
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal wMsg&, ByVal wParam&, lParam As Any)
```

```
' 指定されたウィンドウの位置およびサイズを変更
Declare Function Api_MoveWindow& Lib "user32" Alias "MoveWindow" (ByVal hWnd&, ByVal X&,
ByVal Y&, ByVal nWidth&, ByVal nHeight&, ByVal bRepaint&)

#define CB_SHOWDROPDOWN &H14F              'コンボボックスのリストボックスの表示または非表示を切り
                                           替える
#define CB_GETITEMHEIGHT &H154             'コンボボックス内の項目の高さを取得する

Var Shared Combo1 As Object
Var Shared Combo2 As Object

Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
Combo2.Attach GetDlgItem("Combo2") : Combo2.SetFontSize 14

Var Shared ItemHeight As Long

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var i As Integer
    Var Ret As Long

    Combo1.SetWindowText "初期設定時"
    Combo2.SetWindowText "項目数変更"

    For i = 1 To 20
        Combo1.AddString "data" & Str$(i)
        Combo2.AddString "data" & Str$(i)
    Next i

    'コンボボックスコントロールの項目の高さを取得
    ItemHeight = Api_SendMessage(Combo2.GethWnd, CB_GETITEMHEIGHT, 0, 0)

    SetMousePointer 0
    Ret = Api_SendMessage(Combo1.GethWnd, CB_SHOWDROPDOWN, 1, ByVal 0)
End Sub

'=====================================
'=
'=====================================
Declare Sub Combo1_Change edecl ()
Sub Combo1_Change()
    Var CbNewHeight As Long
    Var Ret As Long

    'コンボボックスコントロールの高さを設定  6項目を表示
    '("+2"はドロップダウンリスト部の上下の境界線の高さ分)
    CbNewHeight = Combo2.GetHeight + ItemHeight * (Combo1.GetCursel + 1 ) + 2

    'コンボボックスコントロール(ウィンドウ)の位置とサイズを変更
    Ret = Api_MoveWindow(Combo2.GethWnd, Combo2.GetLeft, Combo2.GetTop,
Combo2.GetWidth, CbNewHeight, 1)

    SetMousePointer 0
    Ret = Api_SendMessage(Combo2.GethWnd, CB_SHOWDROPDOWN, 1, ByVal 0)
    Combo2.SetFocus
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

コンボボックスの項目表示数を指定

コンボボックスのリスト項目表示数を指定します。
SendMessage ウィンドウにメッセージを送信
MoveWindow 指定されたウィンドウの位置およびサイズを変更
GetWindowRect ウィンドウの座標をスクリーン座標系で取得
ScreenToClient マウスカーソルの現在の位置に相当するスクリーン座標を取得し、クライアント座標に変換
CB_SHOWDROPDOWN(&H14F) コンボボックスのリスト部、表示・非表示切替
CB_GETITEMHEIGHT(&H154) コンボボックス内の項目の高さを取得

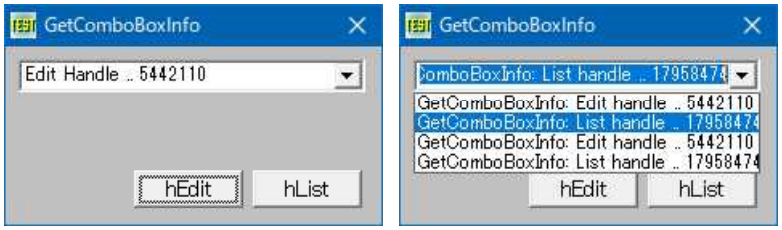



```
'=======================================
'= コンボボックスの項目表示数を指定
'=    MoveWindow3.bas
'=======================================
#include "Windows.bi"

#define CB_GETITEMHEIGHT &H154              'コンボボックス内の項目の高さを取得する
#define CB_SHOWDROPDOWN &H14F               'コンボボックスのリストボックスの表示または非表示を切り替える

Type POINTAPI
   x    As Long
   y    As Long
End Type

Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

' ウィンドウにメッセージを送信
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal wMsg&, ByVal wParam&, lParam As Any)
' 指定されたウィンドウの位置およびサイズを変更
Declare Function Api_MoveWindow& Lib "user32" Alias "MoveWindow" (ByVal hWnd&, ByVal X&, ByVal Y&, ByVal nWidth&, ByVal nHeight&, ByVal bRepaint&)

' ウィンドウの座標をスクリーン座標系で取得
Declare Function Api_GetWindowRect& Lib "user32" Alias "GetWindowRect" (ByVal hWnd&, lpRect As RECT)

' マウスカーソルの現在の位置に相当するスクリーン座標を取得し、クライアント座標に変換
Declare Function Api_ScreenToClient& Lib "user32" Alias "ScreenToClient" (ByVal hWnd&, lpPoint As POINTAPI)

Var Shared Combo1 As Object
Var Shared Text1 As Object
Var Shared Edit1 As Object
Var Shared Button1 As Object

Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=======================================
'=
'=======================================
Declare Sub MainForm_Start edecl ()
```

```
Sub MainForm_Start()
    For i = 1 To 20
        Combo1.AddString "Item-" & Trim$(Str$(i))
    Next
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var pt As POINTAPI
    Var rc As RECT
    Var cWidth As Long
    Var newHeight As Long
    Var oldMapMode As Long
    Var numItemsToDisplay As Long
    Var itemHeight As Long

    numItemsToDisplay = Val(Edit1.GetWindowText)
    If numItemsToDisplay > 20 Then Edit1.SetWindowText "" : Exit Sub

    Text1.SetWindowText "Items displayed = " & Trim$(Str$(numItemsToDisplay))

    oldMapMode = GetMapMode
    SetMapMode 1

    cWidth = Combo1.GetWidth

    itemHeight = Api_SendMessage(Combo1.GethWnd, CB_GETITEMHEIGHT, 0, ByVal 0)

    newHeight = itemHeight * (numItemsToDisplay + 2)

    Ret = Api_GetWindowRect(Combo1.GethWnd, rc)

    pt.x = rc.Left
    pt.y = rc.Top

    Ret = Api_ScreenToClient(GethWnd, pt)

    Ret = Api_MoveWindow(Combo1.GethWnd, pt.x, pt.y, Combo1.GetWidth, newHeight, True)

    Ret = Api_SendMessage(Combo1.GethWnd, CB_SHOWDROPDOWN, True, ByVal 0)

    SetMapMode oldMapMode
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

コンボボックスの情報を取得

GetComboBoxInfo 指定したコンボボックスに関する情報を取得
SendMessage ウィンドウにメッセージを送信
SetWindowText ウィンドウのタイトルを変更

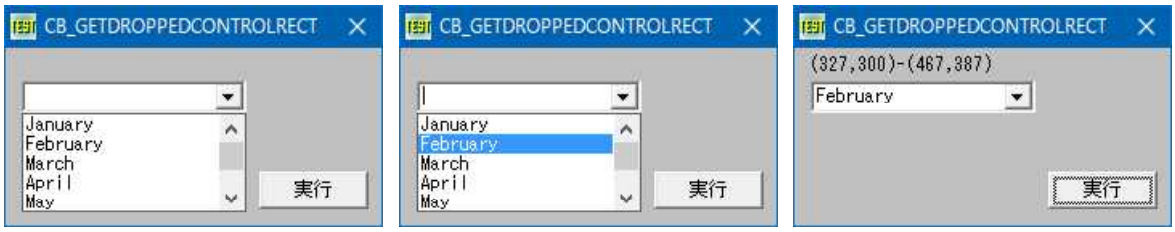



```
'=====================================
'= コンボボックスの情報を取得
'=    (GetComboBoxInfo.bas)
'=====================================
#include "Windows.bi"

#define CB_ADDSTRING &H143                  'コンボボックスのリストボックスに文字列を追加する
#define CB_SETITEMDATA &H151                'アプリケーション定義の値を設定する

Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

Type COMBOBOXINFO
   cbSize       As Long
   rcItem       As RECT
   rcButton     As RECT
   stateButton  As Long
   hwndCombo    As Long
   hwndEdit     As Long
   hwndList     As Long
End Type

' 指定したコンボボックスに関する情報を取得
Declare Function Api_GetComboBoxInfo& Lib "user32" Alias "GetComboBoxInfo" (ByVal
hwndCombo&, pcbi As COMBOBOXINFO)

' ウィンドウにメッセージを送信
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

' ウィンドウのタイトルを変更
Declare Function Api_SetWindowText& Lib "user32" Alias "SetWindowTextA" (ByVal hWnd&,
ByVal lpString$)

Var Shared Combo1 As Object
Var Shared Button1 As Object
Var Shared Button2 As Object

Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 12
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var cbi As COMBOBOXINFO
    Var hEdit As Long
    Var Ret As Long

    cbi.cbSize = Len(cbi)
    Ret = Api_GetComboBoxInfo(Combo1.GethWnd, cbi)

    hEdit = cbi.hwndEdit

    Ret = Api_SetWindowText(hEdit, "Edit Handle .." & Str$(hEdit))
```

```
    'Edit部のハンドルを追加
    Ret = Api_SendMessage(Combo1.GethWnd, CB_ADDSTRING, 0, "GetComboBoxInfo: Edit
handle .." & Str$(hEdit))
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var cbi As COMBOBOXINFO
    Var hList As Long
    Var Ret As Long

    cbi.cbSize = Len(cbi)
    Ret = Api_GetComboBoxInfo(Combo1.GethWnd, cbi)

    hList = cbi.hwndList

    'List部のハンドルを追加
    Ret = Api_SendMessage(Combo1.GethWnd, CB_ADDSTRING, 0, "GetComboBoxInfo: List
handle .." & Str$(hList))
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

コンボボックスのドロップダウンリスト座標

SendMessage 指定のウィンドウにメッセージを送る
CB_GETDROPPEDCONTROLRECT(&H152) コンボボックスのドロップダウンリストボックスの長方形を取得する
CB_SHOWDROPDOWN(&H14F) コンボボックスのリストボックスの表示または非表示を切り替える




```
'=====================================
'= コンボボックスのドロップダウンリスト座標
'=    (CB_GETDROPPEDCONTROLRECT.bas)
'=====================================
#include "Windows.bi"

Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

' ウィンドウにメッセージを送信
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

#define CB_GETDROPPEDCONTROLRECT &H152      'コンボボックスのドロップダウンリストボックスの長方形を取
                                            得する
```

```
#define CB_SHOWDROPDOWN &H14F                    'コンボボックスのリストボックスの表示または非表示を切り替える

Var Shared Combo1 As Object
Var Shared Text1 As Object
Var Shared Button1 As Object

Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 12
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 12
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 12

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var mm(11) As String
    Var Ret As Long

    For i = 0 To 11
        Read mm(i)
        Combo1.AddString mm(i)
    Next

    'コンボボックスドロップダウン表示
    SetMousePointer 0
    Ret = Api_SendMessage(Combo1.GethWnd, CB_SHOWDROPDOWN, 1, ByVal 0)

    Data "January ", "February", "March", "April", "May", "June"
    Data "July", "August", "September", "October", "November", "December"
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var rct As RECT
    Var Ret As Long

    ' コンボボックスのドロップダウンリスト座標を取得
    Ret = Api_SendMessage(Combo1.GethWnd, CB_GETDROPPEDCONTROLRECT, CLng(0), rct)

    '結果を表示
    Text1.SetWindowText "(" & Trim$(Str$(rct.Left)) & "," & Trim$(Str$(rct.Top)) & ")-" &
"(" & Trim$(Str$(rct.Right)) & "," & Trim$(Str$(rct.Bottom)) & ")"
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

コンボボックスのドロップダウンリスト幅を設定(Ⅰ)

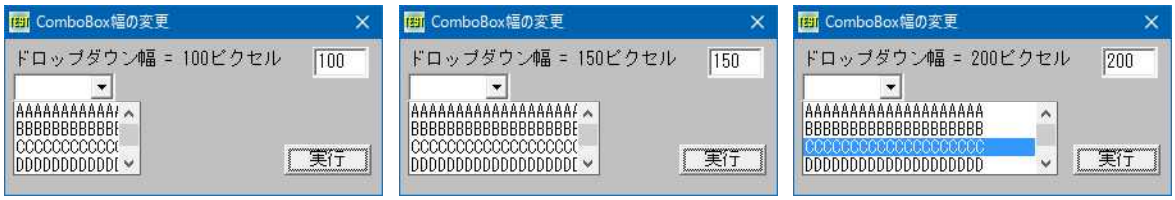
コンボボックスのドロップダウンリスト幅を設定します。
SendMessage ウィンドウにメッセージを送信
CB_SETDROPPEDWIDTH(&H160) コンボボックスのドロップダウンリスト幅を設定(Ⅰ)


```
'=====================================
'=  コンボボックスリストダウン幅を変更
'=    (ComboWidth.bas)
'=====================================
#include "Windows.bi"

' ウィンドウにメッセージを送信。この関数は、指定したウィンドウのウィンドウプロシージャが処理を終了するまで制御を返さない
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal wMsg&, ByVal wParam&, ByVal lParam&)

#define CB_SETDROPPEDWIDTH &H160          'コンボボックスのリスト幅を変更

Var Shared Combo1 As Object
Var Shared Button1 As Object
Var Shared Button2 As Object

Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var i As Long

    For i = 1 To 10
        Combo1.AddString String$(20, Chr$(&H40 + i))
    Next
End Sub

'=====================================
'=  幅を広げる
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var nWidth As Long
    Var Ret As Long

    'ドロップダウンリスト幅を設定(2倍)
    nWidth = CLng(Combo1.GetWidth * 2)
    Ret = Api_SendMessage(Combo1.GethWnd, CB_SETDROPPEDWIDTH, nWidth, ByVal CLng(0))
End Sub

'=====================================
'=  幅を戻す
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var nWidth As Long
    Var Ret  As Long

    'ドロップダウンリスト幅を設定
    nWidth = CLng(Combo1.GetWidth)
```

```
    Ret = Api_SendMessage(Combo1.GethWnd, CB_SETDROPPEDWIDTH, nWidth, ByVal CLng(0))
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

コンボボックスのドロップダウンリスト幅を設定（II）

コンボボックスのドロップダウンリスト幅を設定します。
SendMessage ウィンドウにメッセージを送信
CB_GETLBTEXTLEN(&H149) コンボボックスのリストボックス文字列の長さを取得
CB_GETDROPPEDWIDTH(&H15F) コンボボックスのリスト幅を取得
CB_SHOWDROPDOWN(&H14F) コンボボックスのリストボックスの表示または非表示を切り替える
CB_SETDROPPEDWIDTH(&H160) コンボボックスのリスト幅を変更

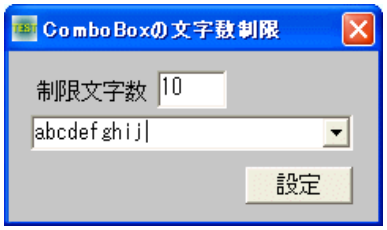



```
'=====================================
'=  コンボボックスのドロップダウンリスト幅を設定（II）
'=    (ComboWidth2.bas)
'=====================================
#include "Windows.bi"

' ウィンドウにメッセージを送信。この関数は、指定したウィンドウのウィンドウプロシージャが処理を終了するまで制御を返さない
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal wMsg&, ByVal wParam&, ByVal lParam&)

#define CB_GETLBTEXTLEN &H149              'コンボボックスのリストボックス文字列の長さを取得
#define CB_GETDROPPEDWIDTH &H15F           'コンボボックスのリスト幅を取得
#define CB_SHOWDROPDOWN &H14F              'コンボボックスのリストボックスの表示または非表示を切り替える
#define CB_SETDROPPEDWIDTH &H160           'コンボボックスのリスト幅を変更

Var Shared Combo1 As Object
Var Shared Edit1 As Object
Var Shared Text1 As Object
Var Shared Button1 As Object

Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var i As Long

    For i = 1 To 10
        Combo1.AddString String$(20, Chr$(&H40 + i))
    Next
End Sub
```

```
'=====================================
'=  幅を広げる
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var cwidth As Long
    Var NewDropDownWidth As Long

    If Val(Edit1.GetWindowText) Then
        NewDropDownWidth = Val(Edit1.GetWindowText)

        Ret = Api_SendMessage(Combo1.GethWnd, CB_SETDROPPEDWIDTH, NewDropDownWidth,
ByVal 0)

        cwidth = Api_SendMessage(Combo1.GethWnd, CB_GETDROPPEDWIDTH, 0, ByVal 0)
        Text1.SetWindowText "ドロップダウン幅 = " & Trim$(Str$(cwidth)) & "ピクセル"

        Ret = Api_SendMessage(Combo1.GethWnd, CB_SHOWDROPDOWN, True, ByVal 0)
   End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

コンボボックスの文字数制限

コンボボックスの入力文字数を制限します。(プロパティ:ドロップダウン)
SendMessage ウィンドウにメッセージを送信
FindWindowEx クラス名、キャプションを与えてウィンドウのハンドルを取得
SetWindowText ウィンドウのタイトルを変更

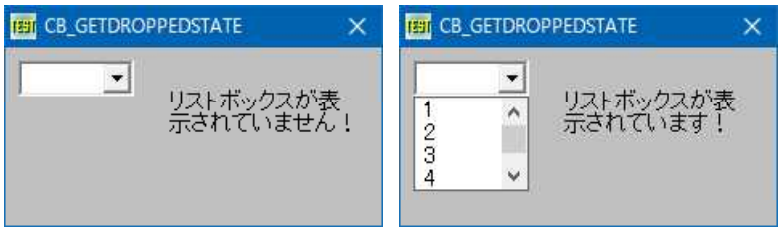



```
'=====================================
'= コンボボックスの文字数制限
'=====================================
#include "Windows.bi"

' ウィンドウにメッセージを送信
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

' クラス名 、または キャプションを与えてウィンドウのハンドルを取得
Declare Function Api_FindWindowEx& Lib "user32" Alias "FindWindowExA" (ByVal
hWndParent&, ByVal hWndChildAfter&, ByVal lpszClass$, ByVal lpszWindow$)

' ウィンドウのタイトルを変更
Declare Function Api_SetWindowText& Lib "user32" Alias "SetWindowTextA" (ByVal hWnd&,
ByVal lpString$)

#define EM_LIMITTEXT &HC5                    'エディットコントロール内のテキストの文字数を制限する

Var Shared Combo1 As Object
Var Shared Edit1 As Object
```

```
Var Shared Text1 As Object
Var Shared Button1 As Object

Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hWnd As Long
    Var mLen As Long
    Var Ret As Long

    hWnd = Api_FindWindowEx(Combo1.GethWnd, 0, ByVal 0, ByVal 0)
    mLen = Val(Edit1.GetWindowText)

    If hWnd <> 0 Then
        If mLen > 0 Then
            Ret = Api_SetWindowText(hWnd, "")
            Ret = Api_SendMessage(hWnd, EM_LIMITTEXT, mLen, 0)
        Else
            Ret = Api_SendMessage(hWnd, EM_LIMITTEXT, 0, 0)
        End If
    End If

    Ret = Api_SetWindowText(GethWnd, Trim$(Str$(mLen)) & "文字に制限")
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

コンボボックスのリスト表示状態の判断

SendMessage ウィンドウにメッセージを送信
CB_GETDROPPEDSTATE(&H157) コンボボックスのリストボックスが表示されている状態かどうかを判断




```
'=====================================
'= コンボボックスのリスト表示状態の判断
'=    (CB_GETDROPPEDSTATE.bas)
'=====================================
#include "Windows.bi"

' ウィンドウにメッセージを送信
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

#define CB_GETDROPPEDSTATE &H157          'コンボボックスのリストボックスが表示されている状態かどうかを判断する
```

```
Var Shared Combo1 As Object
Var Shared Text1 As Object
Var Shared Timer1 As Object

Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Timer1.Attach GetDlgItem("Timer1")

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    For i% = 1 To 20
        Combo1.AddString Str$(i%)
    Next

    Timer1.SetInterval 10
    Timer1.Enable -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var Ret As Long

    Ret = Api_SendMessage(Combo1.GethWnd, CB_GETDROPPEDSTATE, 0, 0)

    If Ret Then
        Text1.SetWindowText "リストボックスが表示されています！"
    Else
        Text1.SetWindowText "リストボックスが表示されていません！"
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

コンボボックスのリスト表示・非表示

SendMessage ウィンドウにメッセージを送信
CB_SHOWDROPDOWN(&H14F) コンボボックスのリストボックスの表示または非表示を切り替える

Windows 10

```
'=====================================
'= コンボボックスのリスト表示・非表示
'=    (CB_SHOWDROPDOWN.bas)
'=====================================
#include "Windows.bi"
```

```
' ウィンドウにメッセージを送信
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

#define CB_SHOWDROPDOWN &H14F                    'コンボボックスのリストボックスの表示または非表示を切り替える

Var Shared Combo1 As Object
Var Shared Button1 As Object
Var Shared Button2 As Object

Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    For i% = 1 To 20
        Combo1.AddString Str$(i%)
    Next
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    Ret = Api_SendMessage(Combo1.GethWnd, CB_SHOWDROPDOWN, True, ByVal 0)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Ret As Long

    Ret = Api_SendMessage(Combo1.GethWnd, CB_SHOWDROPDOWN, False, ByVal 0)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

コンボボックスのロケール情報を取得

コンボボックスのロケール情報を取得します。リストボックスの場合は、LB_GETLOCALEを使います。
SendMessage ウィンドウにメッセージを送信

言語	言語ID	言語グループ
英語(既定)	0409	西ヨーロッパと米国
フランス語	040c	西ヨーロッパと米国
スペイン語	0c0a	西ヨーロッパと米国
イタリア語	0410	西ヨーロッパと米国
スウェーデン語	041D	西ヨーロッパと米国
オランダ語	0413	西ヨーロッパと米国
ポルトガル語(ブラジル)	0416	西ヨーロッパと米国
フィンランド語	040b	西ヨーロッパと米国
ノルウェー語	0414	西ヨーロッパと米国
デンマーク語	0406	西ヨーロッパと米国
ハンガリー語	040e	中央ヨーロッパ
ポーランド語	0415	中央ヨーロッパ
ロシア語	0419	キリル言語
チェコ語	0405	中央ヨーロッパ
ギリシャ語	0408	ギリシャ語
ポルトガル語	0816	西ヨーロッパと米国
トルコ語	041f	トルコ語
日本語	0411	日本語
韓国語	0412	韓国語
ドイツ語	0407	西ヨーロッパと米国
簡体字中国語	0804	簡体字中国語
繁体字中国語	0404	繁体字中国語
アラビア語	0401	アラビア語
ヘブライ語	040d	ヘブライ語

```
'=====================================
'= コンボボックスのロケール情報を取得
'=    (CbGetLocale.bas)
'=====================================
#include "Windows.bi"

' ウィンドウにメッセージを送信
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

#define CB_SHOWDROPDOWN &H14F                'コンボボックスのリストボックスの表示または非表示を切り
                                             替える
#define CB_GETLOCALE 346                     '現在のコンボボックスのロケールを取得する(&H15A)
#define LB_GETLOCALE &H1A6                   '現在のリストボックスのロケールを取得する

Var Shared Combo1 As Object
Var Shared Text1 As Object

Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var i As Integer
    Var Ret As Long

    For i = 1 To 12
        Combo1.AddString Str$(i)
    Next i

    SetMousePointer 0
    Ret = Api_SendMessage(Combo1.GethWnd, CB_SHOWDROPDOWN, 1, ByVal 0)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var LocaleInfo As Long
```

```
    'ロケール情報を取得
    LocaleInfo = Api_SendMessage(Combo1.GethWnd, CB_GETLOCALE, 0, ByVal CLng(0))

    '取得したロケールIDから言語IDを取り出し表示
    Text1.SetWindowtext "言語ID:" & Hex$(LocaleInfo And &H7FFF)
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

コンボボックスをアニメーションで開く

「コンボボックスをアニメーションで開く」の設定と解除
SystemParametersInfo システム全体に関するパラメータを取得・設定

左:アニメーションオープン解除　　右:アニメーションオープン設定
Windows 7 で試したところ「AnimeOn」は効きませんでした。(Vistaではどうなんでしょう？)2010-02-28

Windows10での例（アニメーションで開くことが確認できました）

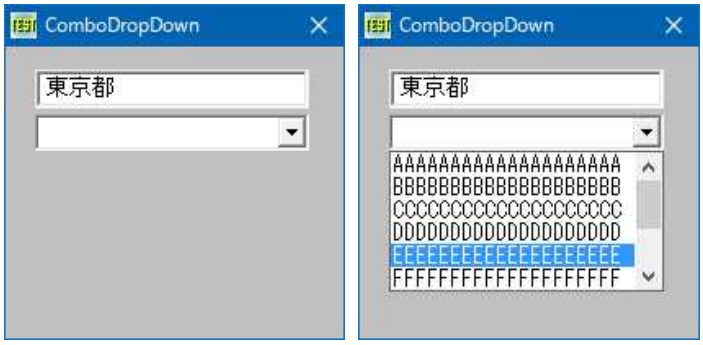

```
'=======================================
'= コンボボックスをアニメーションで開く
'=    (ComboSlideOpen.bas)
'=======================================
#include "Windows.bi"

#define SPI_SETCOMBOBOXANIMATION &H1005    'コンボボックスを開くときアニメーション効果を有効・無効に設定

#define SPIF_UPDATEINIFILE &H1             'ユーザープロファイルの更新を指定する定数の宣言
#define SPIF_SEndWININICHANGE &H2          '全てのアプリケーションに通知して更新する
#define SPIF_SEndCHANGE SPIF_SEndWININICHANGE

' システム全体に関するパラメータを取得・設定
Declare Function Api_SystemParametersInfo& Lib "user32" Alias "SystemParametersInfoA"
(ByVal uiAction&, ByVal uiParam&, pvParam As Any, ByVal fWinIni&)

Var Shared Combo1 As Object
Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
```

```
'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var Ret As Long

    For i = 1000 To 1140
        Combo1.AddString Str$(i)
    Next

    ' コンボボックスのアニメーション効果の有効･無効を設定
    Ret = Api_SystemParametersInfo(SPI_SETCOMBOBOXANIMATION, 0, ByVal 0,
SPIF_UPDATEINIFILE Or SPIF_SEndCHANGE)
End Sub

'=====================================
'=
'=====================================
Declare Sub Radio1_on edecl ()
Sub Radio1_on()
    Var Ret As Long

    ' コンボボックスのアニメーション効果の有効･無効を設定
    Ret = Api_SystemParametersInfo(SPI_SETCOMBOBOXANIMATION, 0, ByVal 0,
SPIF_UPDATEINIFILE Or SPIF_SEndCHANGE)
End Sub

'=====================================
'=
'=====================================
Declare Sub Radio2_on edecl ()
Sub Radio2_on()
    Var Ret As Long

    ' コンボボックスのアニメーション効果の有効･無効を設定
    Ret = Api_SystemParametersInfo(SPI_SETCOMBOBOXANIMATION, 0, ByVal 1,
SPIF_UPDATEINIFILE Or SPIF_SEndCHANGE)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

コンボボックスをドロップダウンで開く

コンボボックスをドロップダウンで開く
SendMessage ウィンドウにメッセージを送信
CB_SHOWDROPDOWN コンボボックスのリストボックスの表示または非表示を切り替える

コンボボックスの▼をクリックして開き、該当行を選択するのは面倒です。
例では、EditBoxに文字を入力し[Enter]を押下すると、カーソルはComboBoxがドロップダウン状態で移動します。
F-Basicの場合、SendMessageの前にSetMousePointer 0(マウス砂時計解除)を入れないと目的動作ができないようです。



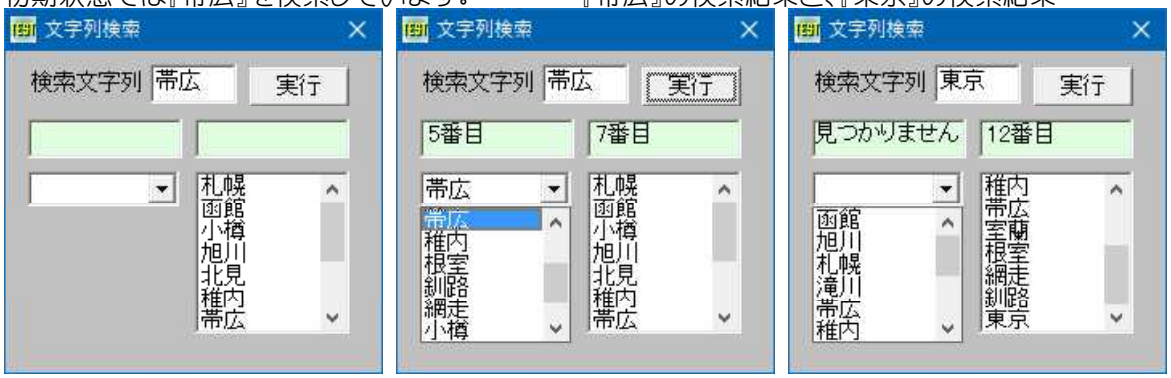

```
'=====================================
'= コンボボックスをドロップダウンで開く
'=    (ComboDropdown.bas)
'=====================================
#include "Windows.bi"

' ウィンドウにメッセージを送信。この関数は、指定したウィンドウのウィンドウプロシージャが処理を終了するまで制御を返さない
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal wMsg&, ByVal wParam&, ByVal lParam&)

#define CB_SHOWDROPDOWN &H14F                'コンボボックスのリストボックスの表示または非表示を切り替える

Var Shared Edit1 As Object
Var Shared Combo1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    For i = 1 To 10
        Combo1.AddString String$(20, Chr$(&H40 + i))
    Next
    Edit1.SetFocus
End Sub

'=====================================
'=
'=====================================
Declare Sub Edit1_Change edecl ()
Sub Edit1_Change()
    Var Epos As Integer
    Var Ret As Long

    ED$ = GetDlgItemText("Edit1")
    Epos = InStr(ED$, Chr$(13,10))
    If Epos <> 0 Then
        ED$ = Mid$(ED$, 1, Epos - 1) & Mid$(ED$, Epos + 2)
        Edit1.SetWindowText ED$
        Combo1.SetFocus

        'マウス砂時計解除
        SetMousePointer 0
        Ret = Api_SendMessage(Combo1.GethWnd, CB_SHOWDROPDOWN, 1, ByVal 0)
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
```

```
Wend
Stop
End
```

コンボ(リスト)ボックス内の文字列検索

コンボボックスのリストボックス内、およびリストボックス内の文字列を検索します。
SendMessage ウィンドウにメッセージを送る関数
CB_FINDSTRINGEXACT(&H158) コンボボックスに対して
LB_FINDSTRINGEXACT(&H1A2) リストボックスに対して

初期状態では『帯広』を検索しています。　　『帯広』の検索結果と、『東京』の検索結果

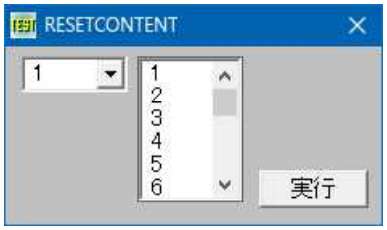


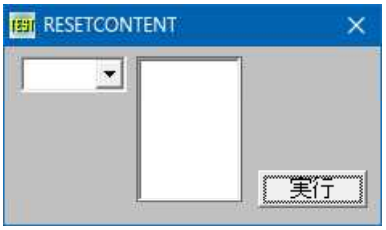

```
'=======================================
'= コンボ(リスト)ボックス内の文字列検索
'=    (FindString.bas)
'=======================================
#include "Windows.bi"

' ウィンドウにメッセージを送る関数の宣言
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

#define CB_FINDSTRINGEXACT &H158          'コンボボックスのリスト内でプリフィックス文字を探す
#define CB_ERR (-1)                       'エラー
#define CB_SHOWDROPDOWN &H14F             'コンボボックスドロップダウンで開く
#define LB_FINDSTRINGEXACT &H1A2          'リストボックスでプリフィックス文字を探す

Var Shared Combo1 As Object
Var Shared List1 As Object
Var Shared Edit1 As Object
Var Shared Button1 As Object
Var Shared Text(2) As Object

Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
For i = 0 To 2
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1))) : Text(i).SetFontSize 14
Next i
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=======================================
'=
'=======================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Combo1.AddString "函館" : List1.AddString "札幌"
    Combo1.AddString "旭川" : List1.AddString "函館"
    Combo1.AddString "札幌" : List1.AddString "小樽"
    Combo1.AddString "滝川" : List1.AddString "旭川"
    Combo1.AddString "帯広" : List1.AddString "北見"
    Combo1.AddString "稚内" : List1.AddString "稚内"
    Combo1.AddString "根室" : List1.AddString "帯広"
```

```
    Combo1.AddString "釧路" : List1.AddString "室蘭"
    Combo1.AddString "網走" : List1.AddString "根室"
    Combo1.AddString "小樽" : List1.AddString "網走"
    Combo1.AddString "北見" : List1.AddString "釧路"
    Combo1.AddString "室蘭" : List1.AddString "東京"

    Edit1.SetWindowText "帯広"
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Start As Long
    Var cFinded As Long
    Var lFinded As Long
    Var Prefix As String
    Var cIdx As Integer
    Var lIdx As Integer

    Start = (-1)                                '検索する条件を全範囲に設定
    Prefix = GetDlgItemText("Edit1")            '検索する文字列を指定

    '指定した文字列を検索
    cFinded = Api_SendMessage(Combo1.GethWnd, CB_FINDSTRINGEXACT, Start, Prefix)
    lFinded = Api_SendMessage(List1.GethWnd, LB_FINDSTRINGEXACT, Start, Prefix)

    '検索した項目を選択
    cIdx = cFinded + 1
    lIdx = lFinded + 1

    If cFinded = CB_ERR Then
        Text(1).SetWindowText "見つかりません"
    Else
        Text(1).SetWindowText Str$(cIdx) & "番目"
    End If

    If lFinded = CB_ERR Then
        Text(2).SetWindowText "見つかりません"
    Else
        Text(2).SetWindowText Str$(lIdx) & "番目"
    End If

    SetMousePointer 0
    Ret = Api_SendMessage(Combo1.GethWnd, CB_SHOWDROPDOWN, 1, ByVal 0)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

コンボ(リスト)ボックスの項目を全て削除

SendMessage ウィンドウにメッセージを送信
CB_RESETCONTENT(&H14B) コンボボックスのリストボックスからすべての項目を除去
LB_RESETCONTENT(&H184) リストボックスからすべての項目を除去



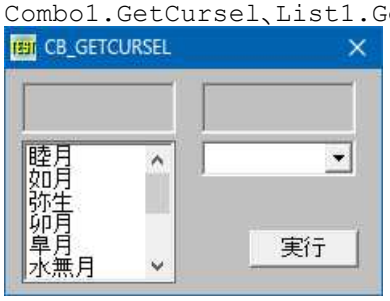



```
'=====================================
'= コンボ(リスト)ボックスの項目を全て削除
'=    (RESETCONTENT.bas)
'=====================================
#include "Windows.bi"

' ウィンドウにメッセージを送信
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

#define CB_RESETCONTENT &H14B              'コンボボックスのリストボックスからすべての項目を除去
#define LB_RESETCONTENT &H184              'リストボックスからすべての項目を除去する

Var Shared Combo1 As Object
Var Shared List1 As Object
Var Shared Button1 As Object

Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
Combo1.SetWindowSize 66, 96
List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
List1.SetWindowSize 66, 96
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    For i% = 1 To 20
        Combo1.AddString Str$(i%)
        List1.AddString Str$(i%)
    Next
    Combo1.SetCursel 0
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    'コンボボックス･リストボックスの項目をすべて削除
    Ret = Api_SendMessage(Combo1.GethWnd, CB_RESETCONTENT, 0, ByVal CLng(0))
    Ret = Api_SendMessage(List1.GethWnd, LB_RESETCONTENT, 0, ByVal CLng(0))
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

コンボ(リスト)ボックスの選択項目番号を取得

SendMessage ウィンドウにメッセージを送信
CB_GETCURSEL(&H147) コンボボックスのリストボックス内で選択された項目のインデックスを取得する
CB_ERR(-1) エラー
LB_GETCURSEL(&H188) リストボックス内で選択された項目のインデックスを取得する
LB_ERR(-1) エラー

Combo1.GetCursel、List1.GetCurselと同じです。



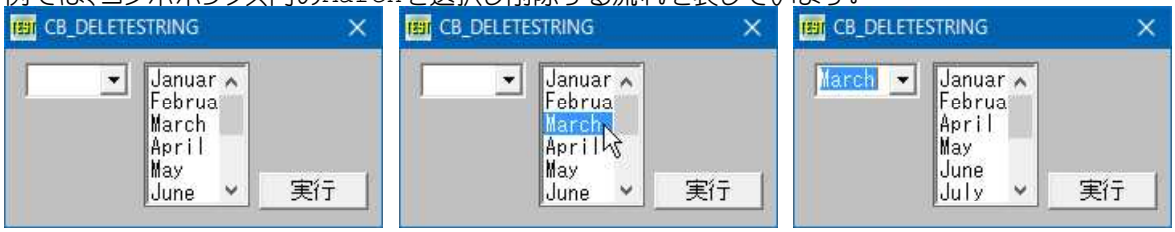

```
'=======================================
'= コンボ(リスト)ボックスの選択項目番号を取得
'=    (CB_GETCURSEL.bas)
'=======================================
#include "Windows.bi"

' ウィンドウにメッセージを送信
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

#define CB_GETCURSEL &H147                  'コンボボックスのリストボックス内で選択された項目のインデックスを取得する
#define CB_ERR (-1)                         'エラー

#define LB_GETCURSEL &H188                  'リストボックス内で選択された項目のインデックスを取得
#define LB_ERR (-1)                         'エラー

Var Shared Combo1 As Object
Var Shared List1 As Object
Var Shared Text1 As Object
Var Shared Text2 As Object
Var Shared Button1 As Object

Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 12
Text2.Attach GetDlgItem("Text2") : Text2.SetFontSize 12
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var Shared Month(11) As String

'=======================================
'=
'=======================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    For i = 0 To 11
        Read Month(i)
        Combo1.AddString Month(i)
        List1.AddString Month(i)
    Next

    Data "睦月","如月","弥生","卯月","皐月","水無月"
    Data "文月","葉月","長月","神無月","霜月","師走"
End Sub

'=======================================
'=
'=======================================
```

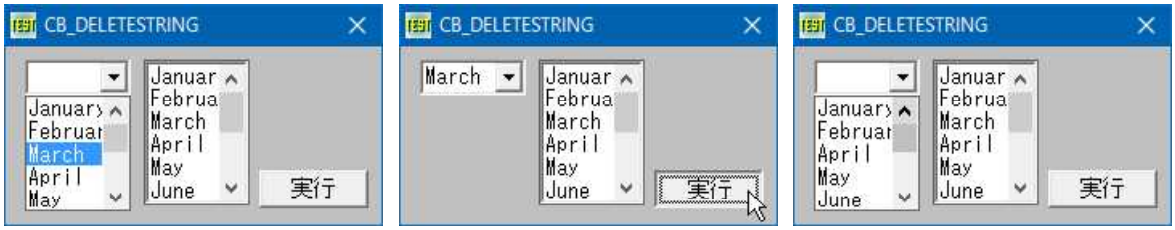
```
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var CbSelIndex As Long
    Var LbSelIndex As Long

    'コンボボックスの選択項目インデックスを取得
    CbSelIndex = Api_SendMessage(Combo1.GethWnd, CB_GETCURSEL, 0, ByVal CLng(0))

    '項目インデックスが取得できたとき
    If CbSelIndex <> CB_ERR Then

        '選択項目インデックスを表示
        Text1.SetWindowText Str$(CbSelIndex)

    '項目インデックスが取得できなかったとき
    Else

        'エラーを表示
        Text1.SetWindowText "項目が選択されていません。"
    End If

    'リストボックスの選択項目インデックスを取得
    LbSelIndex = Api_SendMessage(List1.GethWnd, LB_GETCURSEL, 0, ByVal CLng(0))

    '項目インデックスが取得できたときは
    If LbSelIndex <> LB_ERR Then

        '選択項目インデックスを表示
        Text2.SetWindowText Str$(LbSelIndex)

    '項目インデックスが取得できなかったとき
    Else

        'エラーを表示
        Text2.SetWindowText "項目が選択されていません。"
    End If
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

コンボ(リスト)ボックスの選択項目を削除

SendMessage 指定のウィンドウにメッセージを送る
CB_GETCURSEL(&H147) コンボボックスのリストボックス内で選択された項目のインデックスを取得する
CB_DELETESTRING(&H144) コンボボックスのリストボックス内の文字列を削除する
CB_SHOWDROPDOWN(&H14F) コンボボックスのリストボックスの表示または非表示を切り替える
CB_ERR(-1) エラー
LB_GETCURSEL(&H188) リストボックス内で選択された項目のインデックスを取得する
LB_DELETESTRING(&H182) リストボックス中の文字列を削除する
LB_ERR(-1) エラー

例では、コンボボックス内のMarchを選択し削除する流れを表しています。



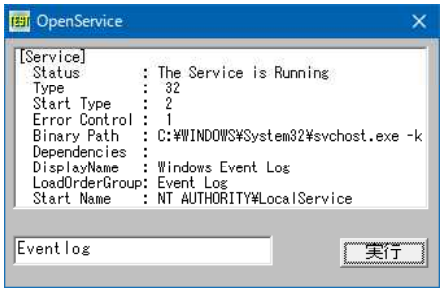



```
'=====================================
'= コンボ(リスト)ボックスの選択項目を削除
'=    (CB_DELETESTRING.bas)
'=====================================
#include "Windows.bi"

' ウィンドウにメッセージを送信
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

#define CB_GETCURSEL &H147                  'コンボボックスのリストボックス内で選択された項目のインデックスを取得
#define CB_DELETESTRING &H144               'コンボボックスのリストボックス内の文字列を削除する
#define CB_SHOWDROPDOWN &H14F               'コンボボックスのリストボックスの表示または非表示を切り替える
#define CB_ERR (-1)                         'エラー

#define LB_GETCURSEL &H188                  'リストボックス内で選択された項目のインデックスを取得
#define LB_DELETESTRING &H182               'リストボックス中の文字列を削除する
#define LB_ERR (-1)                         'エラー

Var Shared Combo1 As Object
Var Shared List1 As Object
Var Shared Button1 As Object

Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
Combo1.SetWindowSize 66, 96
List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
List1.SetWindowSize 66, 96
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var mm(11) As String
    Var Ret As Long

    For i = 0 To 11
        Read mm(i)
        Combo1.AddString mm(i)
        List1.AddString mm(i)
    Next

    'コンボボックスドロップダウン表示
    SetMousePointer 0
    Ret = Api_SendMessage(Combo1.GethWnd, CB_SHOWDROPDOWN, 1, ByVal 0)

    Data "January ", "February", "March", "April", "May", "June"
    Data "July", "August", "September", "October", "November", "December"
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var SelIndex As Long
    Var Ret As Long
```

```
    'コンボボックスの選択項目インデックスを取得
    SelIndex = Api_SendMessage(Combo1.GethWnd, CB_GETCURSEL, 0, ByVal CLng(0))

    '項目インデックスが取得できたとき
    If SelIndex <> CB_ERR Then

        '選択項目を削除
        Ret = Api_SendMessage(Combo1.GethWnd, CB_DELETESTRING, SelIndex, ByVal CLng(0))

        'エディットボックス部をクリア
        Combo1.SetWindowText ""
    End If

    'リストボックスの選択項目インデックスを取得
    SelIndex = Api_SendMessage(List1.GethWnd, LB_GETCURSEL, 0, ByVal CLng(0))

    '項目インデックスが取得できたとき
    If SelIndex <> LB_ERR Then

        '選択項目を削除
        Ret = Api_SendMessage(List1.GethWnd, LB_DELETESTRING, SelIndex, ByVal CLng(0))
    End If
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

サービスの状態および構成を取得(Ⅰ)

CloseServiceHandle サービスコントロールマネージャオブジェクトまたはサービスオブジェクトへの指定されたハンドルを閉じる
QueryServiceStatus 指定されたサービスの現在のステータスを取得
OpenService 既存のサービスのハンドルを開く
OpenSCManager 指定されたコンピュータ上のサービス制御マネージャとの接続を確立し、サービス制御マネージャの指定されたデータベースを開く
QueryServiceConfig 指定されたサービスの構成パラメータを取得
CopyMemory 文字列をコピーする
lstrcpy ある位置から別の位置にメモリブロックを移動



HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services で確認できます。

```
'=======================================
'= サービスの状態および構成を取得
'=    (OpenService.bas)
'=    HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services
'=======================================
#include "Windows.bi"

Type SERVICE_STATUS
    dwServiceType                   As Long
    dwCurrentState                  As Long
    dwControlsAccepted              As Long
```

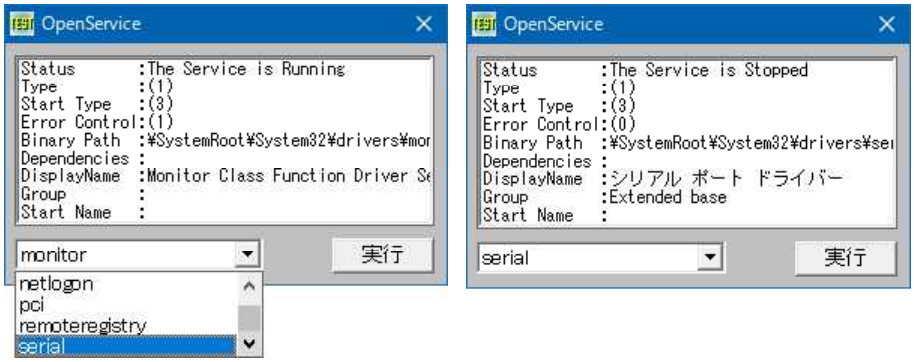
```
    dwWin32ExitCode                 As Long
    dwServiceSpecificExitCode       As Long
    dwCheckPoint                    As Long
    dwWaitHint                      As Long
End Type

Type QUERY_SERVICE_CONFIG
    dwServiceType           As Long
    dwStartType             As Long
    dwErrorControl          As Long
    lpBinaryPathName        As Long
    lpLoadOrderGroup        As Long
    dwTagId                 As Long
    lpDependencies          As Long
    lpServiceStartName      As Long
    lpDisplayName           As Long
End Type

#define SERVICE_ACCEPT_PAUSE_CONTINUE &H2 '
#define SERVICE_ACCEPT_SHUTDOWN &H4       '
#define SERVICE_ACCEPT_STOP &H1           '
#define SERVICE_CONTINUE_PENDING &H5      '
#define SERVICE_PAUSE_PENDING &H6         '
#define SERVICE_PAUSED &H7                '
#define SERVICE_RUNNING &H4               '
#define SERVICE_START_PENDING &H2         '
#define SERVICE_STOP_PENDING &H3          '
#define SERVICE_STOPPED &H1               '
#define SERVICE_INTERROGATE &H80          '
#define SC_MANAGER_CONNECT &H1            '
#define GENERIC_READ -2147483648          '読み込みモード(&H80000000)
#define ERROR_INSUFFICIENT_BUFFER 122     'システム コールに渡されるデータ領域が小さい

' サービスコントロールマネージャオブジェクトまたはサービスオブジェクトへの指定されたハンドルを閉じる
Declare Function Api_CloseServiceHandle& Lib "advapi32" Alias "CloseServiceHandle"
(ByVal hSCObject&)

' 指定されたサービスの現在のステータスを取得
Declare Function Api_QueryServiceStatus& Lib "advapi32" Alias "QueryServiceStatus"
(ByVal hService&, lpServiceStatus As SERVICE_STATUS)

' 既存のサービスのハンドルを開く
Declare Function Api_OpenService& Lib "advapi32" Alias "OpenServiceA" (ByVal
hSCManager&, ByVal lpServiceName$, ByVal dwDesiredAccess&)

' 指定されたコンピュータ上のサービス制御マネージャとの接続を確立し、サービス制御マネージャの指定されたデータベースを開く
Declare Function Api_OpenSCManager& Lib "advapi32" Alias "OpenSCManagerA" (ByVal
lpMachineName$, ByVal lpDatabaseName$, ByVal dwDesiredAccess&)

' 指定されたサービスの構成パラメータを取得
Declare Function Api_QueryServiceConfig& Lib "advapi32" Alias "QueryServiceConfigA"
(ByVal hService&, lpServiceConfig As Byte, ByVal cbBufSize&, pcbBytesNeeded&)

' 文字列をコピーする
Declare Sub CopyMemory Lib "Kernel32" Alias "RtlMoveMemory" (hpvDest As Any, hpvSource As
Any, ByVal cbCopy&)

' ある位置から別の位置にメモリブロックを移動
Declare Function Api_lstrcpy& Lib "Kernel32" Alias "lstrcpyA" (ByVal lpString1$, ByVal
lpString2&)

Var Shared Edit1 As Object
Var Shared List1 As Object
Var Shared Button1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
List1.Attach GetDlgItem("List1") : List1.SetFontSize 12
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
```

```
'====================================
'=
'====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hSCM  As Long
    Var hSVC As Long
    Var ss As SERVICE_STATUS
    Var qsc As QUERY_SERVICE_CONFIG
    Var lBytesNeeded As Long
    Var sTemp As String
    Var pFileName As Long
    Var Ret As Long

    List1.ResetContent

    hSCM = Api_OpenSCManager(ByVal 0, ByVal 0, SC_MANAGER_CONNECT)
    If hSCM = 0 Then
        A% = MessageBox("", "Error", 0, 2)
    End If

    hSVC = Api_OpenService(hSCM, Trim$(Edit1.GetWindowText), GENERIC_READ)
    If hSVC = 0 Then
        A% = MessageBox("", "Error", 0, 2)
        GoTo *CloseHandles
    End If

    Ret = Api_QueryServiceStatus(hSVC, ss)
    If Ret = 0 Then
        A% = MessageBox("", "Error", 0, 2)
        GoTo *CloseHandles
    End If

    Select Case ss.dwCurrentState
        Case SERVICE_STOPPED
            sTemp = "The Service is Stopped"
        Case SERVICE_START_PENDING
            sTemp = "The Service Being Started"
        Case SERVICE_STOP_PENDING
            sTemp = "The Service is in the process of being stopped"
        Case SERVICE_RUNNING
            sTemp = "The Service is Running"
        Case SERVICE_CONTINUE_PENDING
            sTemp = "The Service is in the process of being Continued"
        Case SERVICE_PAUSE_PENDING
            sTemp = "The Service is in the process of being Paused"
        Case SERVICE_PAUSED
            sTemp = "The Service is Paused"
        Case SERVICE_ACCEPT_STOP
            sTemp = "The Service is Stopped"
        Case SERVICE_ACCEPT_PAUSE_CONTINUE
            sTemp = "The Service is "
        Case SERVICE_ACCEPT_SHUTDOWN
            sTemp = "The Service is being Shutdown"
    End Select

    List1.AddString "[Service]"
    List1.AddString "  Status       : " & sTemp
    Dim abConfig(0) As Byte
    Ret = Api_QueryServiceConfig(hSVC, abConfig(0), 0, lBytesNeeded)

    Erase abConfig

    Var abConfig(lBytesNeeded) As Byte
    Ret = Api_QueryServiceConfig(hSVC, abConfig(0), lBytesNeeded, lBytesNeeded)

    CopyMemory qsc, abConfig(0), Len(qsc)

    List1.AddString "  Type         : " & Str$(qsc.dwServiceType)
    List1.AddString "  Start Type   : " & Str$(qsc.dwStartType)
```

```
    List1.AddString "  Error Control : " & Str$(qsc.dwErrorControl)

    sTemp = Space$(255)
    Ret = Api_lstrcpy(sTemp, qsc.lpBinaryPathName)
    List1.AddString "  Binary Path   : " & sTemp

    Ret = Api_lstrcpy(sTemp, qsc.lpDependencies)
    List1.AddString "  Dependencies  : " & sTemp

    Ret = Api_lstrcpy(sTemp, qsc.lpDisplayName)
    List1.AddString "  DisplayName   : " & sTemp

    Ret = Api_lstrcpy(sTemp, qsc.lpLoadOrderGroup)
    List1.AddString "  LoadOrderGroup: " & sTemp

    Ret = Api_lstrcpy(sTemp, qsc.lpServiceStartName)
    List1.AddString "  Start Name    : " & sTemp

*CloseHandles

    'サービスのハンドルをクローズ
    Ret = Api_CloseServiceHandle(hSVC)

    'サービスコントロールマネージャーのハンドルをクローズ
    Ret = Api_CloseServiceHandle(hSCM)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

サービスの状態および構成を取得(II)

OpenService 既存のサービスのハンドルを開く
QueryServiceStatus 指定されたサービスの現在のステータスを取得
OpenSCManager 指定されたコンピュータ上のサービス制御マネージャとの接続を確立し、サービス制御マネージャの指定されたデータベースを開く
QueryServiceConfig 指定されたサービスの構成パラメータを取得
CloseServiceHandle サービスコントロールマネージャオブジェクトまたはサービスオブジェクトへの指定されたハンドルを閉じる
CopyMemory ある位置から別の位置にメモリブロックを移動
lstrcpy 文字列をコピーする




```
'=====================================
'= サービスの状態および構成を取得
'=    (OpenSCManager3.bas)
'=    HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services
'=====================================
```

```
#include "Windows.bi"

Type SERVICE_STATUS
    dwServiceType             As Long
    dwCurrentState            As Long
    dwControlsAccepted        As Long
    dwWin32ExitCode           As Long
    dwServiceSpecificExitCode As Long
    dwCheckPoint              As Long
    dwWaitHint                As Long
End Type

Type QUERY_SERVICE_CONFIG
    dwServiceType             As Long
    dwStartType               As Long
    dwErrorControl            As Long
    lpBinaryPathName          As Long
    lpLoadOrderGroup          As Long
    dwTagId                   As Long
    lpDependencies            As Long
    lpServiceStartName        As Long
    lpDisplayName             As Long
 End Type

#define SERVICE_STOPPED &H1                    'サービスは停止中
#define SERVICE_START_PENDING &H2              'サービスは起動中
#define SERVICE_STOP_PENDING &H3               'サービスは停止するまで待機中
#define SERVICE_RUNNING &H4                    'サービスは実行中
#define SERVICE_CONTINUE_PENDING &H5           '一時停止しているサービスの直前の続行要求が保留
#define SERVICE_PAUSE_PENDING &H6              '実行中のサービスの直前の一時停止要求が保留
#define SERVICE_PAUSED &H7                     'サービスは一時停止
#define SERVICE_ACCEPT_STOP &H1                'サービスを停止
#define SERVICE_ACCEPT_PAUSE_CONTINUE &H2      'サービスを中断、再開
#define SERVICE_ACCEPT_SHUTDOWN &H4            'サービスにいつシステムのシャットダウンが起こるかを通知
#define SERVICE_INTERROGATE &H80               'サービスに現在のステータスを報告
#define GENERIC_READ -2147483648               '読み込みモード(&H80000000)
#define ERROR_INSUFFICIENT_BUFFER 122          'システム コールに渡されるデータ領域が小さい
#define SC_MANAGER_CONNECT &H1                 '

' 既存のサービスのハンドルを開く
Declare Function Api_OpenService& Lib "advapi32" Alias "OpenServiceA" (ByVal
hSCManager&, ByVal lpServiceName$, ByVal dwDesiredAccess&)

' 指定されたサービスの現在のステータスを取得
Declare Function Api_QueryServiceStatus& Lib "advapi32" Alias "QueryServiceStatus"
(ByVal hService&, lpServiceStatus As SERVICE_STATUS)

' 指定されたコンピュータ上のサービス制御マネージャとの接続を確立し、サービス制御マネージャの指定されたデータベースを開く
Declare Function Api_OpenSCManager& Lib "advapi32" Alias "OpenSCManagerA" (ByVal
lpMachineName$, ByVal lpDatabaseName$, ByVal dwDesiredAccess&)

' 指定されたサービスの構成パラメータを取得
Declare Function Api_QueryServiceConfig& Lib "advapi32" Alias "QueryServiceConfigA"
(ByVal hService&, lpServiceConfig As Byte, ByVal cbBufSize&, pcbBytesNeeded&)

' サービスコントロールマネージャオブジェクトまたはサービスオブジェクトへの指定されたハンドルを閉じる
Declare Function Api_CloseServiceHandle& Lib "advapi32" Alias "CloseServiceHandle"
(ByVal hSCObject&)

' ある位置から別の位置にメモリブロックを移動
Declare Sub CopyMemory Lib "Kernel32" Alias "RtlMoveMemory" (Destination As Any, Source
As Any, ByVal Length&)

' 文字列をコピーする
Declare Function Api_lstrcpy& Lib "Kernel32" Alias "lstrcpyA" (ByVal lpszString1$, ByVal
lpszString2&)

Var Shared List1 As Object
Var Shared Combo1 As Object
```

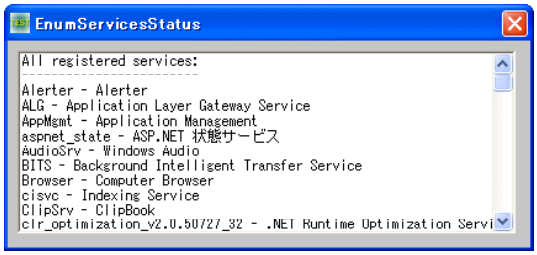
```
Var Shared Button1 As Object

List1.Attach GetDlgItem("List1") : List1.SetFontSize 12
Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'= HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Combo1.AddString "atapi"
    Combo1.AddString "eventlog"
    Combo1.AddString "eventsystem"
    Combo1.AddString "fax"
    Combo1.AddString "monitor"
    Combo1.AddString "netlogon"
    Combo1.AddString "pci"
    Combo1.AddString "remoteregistry"
    Combo1.AddString "serial"
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hSCM  As Long
    Var hSVC As Long
    Var ss As SERVICE_STATUS
    Var qsc As QUERY_SERVICE_CONFIG
    Var Needed As Long
    Var Temp As String
    Var FileName As Long
    Var ServiceName As String
    Var Conf(300) As Byte
    Var Ret As Long

    List1.ResetContent

    hSCM = Api_OpenSCManager(ByVal 0, ByVal 0, SC_MANAGER_CONNECT)
    If hSCM = 0 Then
        A% = MessageBox("1", "Error!", 0, 2)
    End If

    ServiceName = Combo1.GetText(Combo1.GetCursel)
    hSVC = Api_OpenService(hSCM, ServiceName, GENERIC_READ)
    If hSVC = 0 Then
        A% = MessageBox("2", "Error!", 0, 2)
        GoTo *CloseHandles
    End If

    Ret = Api_QueryServiceStatus(hSVC, ss)
    If Ret = 0 Then
        A% = MessageBox("3", "Error!", 0, 2)
        GoTo *CloseHandles
    End If

    Select Case ss.dwCurrentState
        Case SERVICE_STOPPED
            Temp = "The Service is Stopped"
        Case SERVICE_START_PENDING
            Temp = "The Service Being Started"
        Case SERVICE_STOP_PENDING
            Temp = "The Service is in the process of being stopped"
        Case SERVICE_RUNNING
            Temp = "The Service is Running"
        Case SERVICE_CONTINUE_PENDING
            Temp = "The Service is in the process of being Continued"
        Case SERVICE_PAUSE_PENDING
```

```
            Temp = "The Service is in the process of being Paused"
        Case SERVICE_PAUSED
            Temp = "The Service is Paused"
        Case SERVICE_ACCEPT_STOP
            Temp = "The Service is Stopped"
        Case SERVICE_ACCEPT_PAUSE_CONTINUE
            Temp = "The Service is "
        Case SERVICE_ACCEPT_SHUTDOWN
            Temp = "The Service is being Shutdown"
    End Select

    List1.AddString "Status      :" & Temp

    Ret = Api_QueryServiceConfig(hSVC, Conf(0), 0, Needed)
    Ret = Api_QueryServiceConfig(hSVC, Conf(0), Needed, Needed)
    If Ret = 0 Then
        A% = MessageBox("4", "Error =" & Str$(Err), 0, 2)
        GoTo *CloseHandles
    End If

    CopyMemory qsc, Conf(0), Len(qsc)

    List1.AddString "Type        :(" & Trim$(Str$(qsc.dwServiceType)) & ")"
    List1.AddString "Start Type  :(" & Trim$(Str$(qsc.dwStartType)) & ")"
    List1.AddString "Error Control:(" & Trim$(Str$(qsc.dwErrorControl)) & ")"

    Temp = Space$(255)

    Ret = Api_lstrcpy(Temp, qsc.lpBinaryPathName)
    List1.AddString "Binary Path  :" & Temp

    Ret = Api_lstrcpy(Temp, qsc.lpDependencies)
    List1.AddString "Dependencies :" & Temp

    Ret = Api_lstrcpy(Temp, qsc.lpDisplayName)
    List1.AddString "DisplayName  :" & Temp

    Ret = Api_lstrcpy(Temp, qsc.lpLoadOrderGroup)
    List1.AddString "Group       :" & Temp

    Ret = Api_lstrcpy(Temp, qsc.lpServiceStartName)
    List1.AddString "Start Name   :" & Temp

*CloseHandles:
    Ret = Api_CloseServiceHandle(hSVC)
    Ret = Api_CloseServiceHandle(hSCM)
End Sub
'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

サービスを列挙

サービス(参照:http://yougo.ascii24.com/gh/13/001302.html)を列挙します。WindowsNT3.1以降
OpenSCManager サービスマネージャの指定されたデータベースを開く
EnumServicesStatus サービスを列挙
CloseServiceHandle サービスのハンドルを閉じる
lstrcpy 文字列をコピーする

コントロールパネル → 管理ツール → サービス で確認しています。(WindowsXPで確認)
Windows10では動作が停止してしまいました。



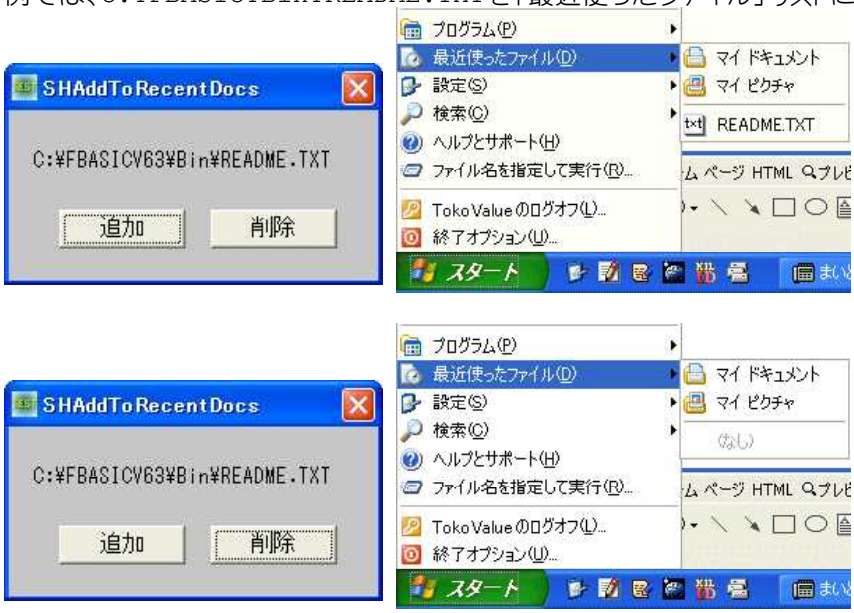

```
'=====================================
'= サービスを列挙
'= WindowsNT3.1以降
'=    (OpenSCManager2.bas)
'=====================================
#include "Windows.bi"

#define ERROR_MORE_DATA 234
#define SERVICE_ACTIVE &H1                    'Active状態のサービスを列挙
#define SERVICE_INACTIVE &H2                  'SERVICE_STOPPED状態のサービスを列挙
#define SC_MANAGER_ENUMERATE_SERVICE &H4      'SCMデータベースにインストールされているサービスを列挙するために必要な
                                              EnumServicesStatus(Ex)の呼び出しを有効に
#define SERVICE_WIN32_OWN_PROCESS &H10        '単体で動作するサービス
#define SERVICE_WIN32_SHARE_PROCESS &H20      '他のサービスと伴に動作するサービス
#define SERVICE_WIN32 (&H10 Or &H20)          '(SERVICE_WIN32_OWN_PROCESS Or
                                              SERVICE_WIN32_SHARE_PROCESS)
#define vbNullString ByVal 0                  '値0の文字列。値0を持つ文字列。空文字列ではない

Type SERVICE_STATUS
    dwServiceType                 As Long
    dwCurrentState                As Long
    dwControlsAccepted            As Long
    dwWin32ExitCode               As Long
    dwServiceSpecificExitCode     As Long
    dwCheckPoint                  As Long
    dwWaitHint                    As Long
End Type

Type ENUM_SERVICE_STATUS
    lpServiceName     As Long
    lpDisplayName     As Long
    ServiceStatus     As SERVICE_STATUS
End Type

' 指定されたコンピュータ上のサービス制御マネージャとの接続を確立し、サービス制御マネージャの指定されたデータベースを開く
Declare Function Api_OpenSCManager& Lib "advapi32" Alias "OpenSCManagerA" (ByVal
lpMachineName$, ByVal lpDatabaseName$, ByVal dwDesiredAccess&)

' サービス制御マネージャ(Service Control Manager:SCM)の指定されたデータベース内のサービスを列挙
Declare Function Api_EnumServicesStatus& Lib "advapi32" Alias "EnumServicesStatusA"
(ByVal hSCManager&, ByVal dwServiceType&, ByVal dwServiceState&, lpServices As Any,
ByVal cbBufSize&, pcbbNeeded&, lpReturned&, lpResumeHandle&)

' サービスコントロールマネージャオブジェクトまたはサービスオブジェクトへの指定されたハンドルを閉じる
Declare Function Api_CloseServiceHandle& Lib "advapi32" Alias "CloseServiceHandle"
(ByVal hSCObject&)

' 文字列をコピーする
Declare Function Api_lstrcpy& Lib "kernel32" Alias "lstrcpyA" (ByVal szDest$, ByVal
szcSource&)

Var Shared List1 As Object

List1.Attach GetDlgItem("List1") : List1.SetFontSize 12

'=====================================
'= Chr$(0)を取り除く
'=====================================
```

```
Declare Function TrimNull(sInput As String) As String
Function TrimNull(sInput As String) As String
    Var ZeroPos As Integer

    ZeroPos = InStr(1, sInput, Chr$(0))
    If ZeroPos > 0 Then
        TrimNull = Left$(sInput, ZeroPos - 1)
    Else
        TrimNull = sInput
    End If
End Function

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var hSCM As Long
    Var ess(261) As ENUM_SERVICE_STATUS
    Var InfoBuffer As Long
    Var ServiceName As String * 250
    Var ByteNeed As Long
    Var ServiceRet As Long
    Var NextUnreadEntry As Long
    Var StructNeed As Long
    Var Ret As Long
    Var i As Long
    Var txt As String

    hSCM = Api_OpenSCManager(vbNullString, vbNullString, SC_MANAGER_ENUMERATE_SERVICE)

    NextUnreadEntry = 0

    Ret = Api_EnumServicesStatus(hSCM, SERVICE_WIN32,
        SERVICE_ACTIVE Or SERVICE_INACTIVE, ByVal 0, 0, ByteNeed, ServiceRet,
NextUnreadEntry)

    StructNeed = ByteNeed / Len(ess(0)) + 1

    InfoBuffer = StructNeed * Len(ess(0))
    NextUnreadEntry = 0

    Ret = Api_EnumServicesStatus(hSCM, SERVICE_WIN32, SERVICE_ACTIVE
                    Or SERVICE_INACTIVE, ess(0), InfoBuffer, ByteNeed, ServiceRet,
                                                        NextUnreadEntry)

    List1.Resetcontent
    List1.AddString "All registered services:"
    List1.AddString "------------------------"

    For i = 0 To ServiceRet - 1
        Ret = Api_lstrcpy(ServiceName, ess(i).lpServiceName)
        txt = TrimNull(ServiceName) & " - "
        Ret = Api_lstrcpy(ServiceName, ess(i).lpDisplayName)
        txt = txt & TrimNull(ServiceName)
        List1.AddString txt
    Next i

    Ret = Api_CloseServiceHandle(hSCM)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

「最近使ったファイル」の追加・削除

「最近使ったファイル」リストの追加と削除を実行します。
SHAddToRecentDocs「最近使ったファイル」のリストにショートカットを追加

例では、C:¥FBASIC¥Bin¥README.TXTを「最近使ったファイル」リストに追加及び削除した状態を示しています。

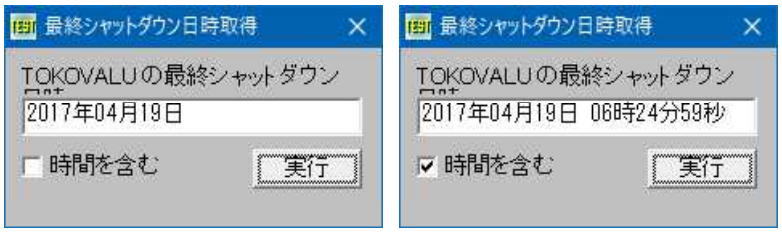



```
'=======================================
'=「最近使ったファイル」の追加・削除
'=    (SHAddToRecentDocs.bas)
'=======================================
#include "Windows.bi"

'「最近使ったファイル」のリストにショートカットを追加
Declare Sub Api_SHAddToRecentDocs Lib "shell32" Alias "SHAddToRecentDocs" (ByVal
uFlags&, pv As Any)

#define SHARD_PIDL &H1
#define SHARD_PATH &H2

Var Shared Text1 As Object
Var Shared Button1 As Object
Var Shared Button2 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var sPath As String

    sPath = "C:¥FBASIC63¥Bin¥README.TXT"
    Api_SHAddToRecentDocs SHARD_PATH, sPath
End Sub

'=======================================
'=
'=======================================
Declare Sub Button2_on edecl ()
Sub Button2_on()

    Api_SHAddToRecentDocs SHARD_PATH, ByVal 0
End Sub
```

```
'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

最終シャットダウン日時を取得

GetTimeZoneInformation タイムゾーン情報を取得
SystemTimeToTzSpecificLocalTime 世界協定時刻(UTC)を、指定したタイムゾーンの対応するローカル時刻に変換
FileTimeToSystemTime ファイルタイムをシステムタイムに変換
RegOpenKeyEx レジストリのキーのハンドルを確保
RegQueryValueEx レジストリの値を取得
RegCloseKey レジストリのハンドルを解放
GetComputerName コンピュータの名前を文字列として取得
lstrlenW (Null文字で終了する)UNICODE文字列の文字数を返す




```
'=====================================
'= 最終シャットダウン日時を取得
'=    (SystemTimeToTzSpecificLocalTime.bas)
'=====================================
#include "Windows.bi"

#define MAX_COMPUTERNAME 16
#define REG_BINARY 3                        'バイナリデータ
#define HKEY_LOCAL_MACHINE -2147483646      'PCを利用するユーザーに共通の設定情報
#define ERROR_SUCCESS &H0                   '正常終了の戻り値を示す
#define STANDARD_RIGHTS_READ &H20000        '(READ_CONTROL)オブジェクトのセキュリティ記述子の読み取り許可

#define KEY_QUERY_VALUE &H1                 'サブキーデータを問い合わせるためのアクセス権
#define KEY_ENUMERATE_SUB_KEYS &H8          'サブキーの列挙を許可
#define KEY_NOTIFY &H10                     '変更の通知を許可
#define SYNCHRONIZE &H100000                '同期をとる

Type SYSTEMTIME
    wYear               As Integer          '年
    wMonth              As Integer          '月(1:1月   2:2月  ...)
    wDayOfWeek          As Integer          '曜(0:日曜  1:月曜 ...)
    wDay                As Integer          '日(1:1日   2:2日  ...)
    wHour               As Integer          '時
    wMinute             As Integer          '分
    wSecond             As Integer          '秒
    wMilliseconds       As Integer          'ミリ秒
End Type

Type FILETIME
    dwLowDateTime       As Long             '下位32ビット値
    dwHighDateTime      As Long             '上位32ビット値
End Type

Type TIME_ZONE_INFORMATION
    Bias                As Long
    StandardName(32)    As Integer
    StandardDate        As SYSTEMTIME
```

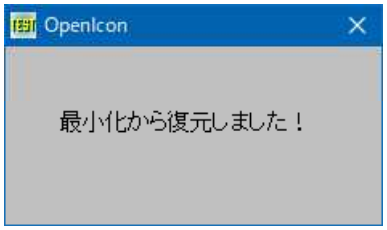
```
    StandardBias      As Long
    DaylightName(32)  As Integer
    DaylightDate      As SYSTEMTIME
    DaylightBias      As Long
End Type

' タイムゾーン情報を取得
Declare Function Api_GetTimeZoneInformation& Lib "kernel32" Alias
"GetTimeZoneInformation" (lpTimeZoneInformation As TIME_ZONE_INFORMATION)

' 世界協定時刻(UTC)を、指定したタイムゾーンの対応するローカル時刻に変換
Declare Function Api_SystemTimeToTzSpecificLocalTime& Lib "kernel32" Alias
"SystemTimeToTzSpecificLocalTime" (lpTimeZoneInformation As TIME_ZONE_INFORMATION,
lpUniversalTime As SYSTEMTIME, lpLocalTime As SYSTEMTIME)

' ファイルタイムをシステムタイムに変換
Declare Function Api_FileTimeToSystemTime& Lib "kernel32" Alias "FileTimeToSystemTime"
(lpFileTime As FILETIME, lpSystemTime As SYSTEMTIME)

' レジストリのキーのハンドルを確保
Declare Function Api_RegOpenKeyEx& Lib "advapi32" Alias "RegOpenKeyExA" (ByVal hKey&,
ByVal lpSubKey$, ByVal ulOptions&, ByVal samDesired&, phkResult&)

' レジストリの値を取得
Declare Function Api_RegQueryValueEx& Lib "advapi32" Alias "RegQueryValueExA" (ByVal
hKey&, ByVal lpvName$, ByVal lpReserved&, lpType&, lpData As Any, lpcbData&)

' レジストリのハンドルを解放
Declare Function Api_RegCloseKey& Lib "advapi32" Alias "RegCloseKey" (ByVal hKey&)

' コンピュータの名前を文字列として取得
Declare Function Api_GetComputerName& Lib "Kernel32" Alias "GetComputerNameA" (ByVal
lpBuffer$, nSize&)

' (Null文字で終了する)UNICODE文字列の文字数を返す
Declare Function Api_lstrlenW& Lib "Kernel32" Alias "lstrlenW" (ByVal lpString&)

Var Shared Edit1 As Object
Var Shared Text1 As Object
Var Shared Check1 As Object
Var Shared Button1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Check1.Attach GetDlgItem("Check1") : Check1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Function GetFileToSystemDate(ft As FILETIME, bIncludeTime As Integer) As String
Function GetFileToSystemDate(ft As FILETIME, bIncludeTime As Integer) As String
    Var buff As String
    Var st As SYSTEMTIME                    'system(UNC)time
    Var lt As SYSTEMTIME                    'local time
    Var tz As TIME_ZONE_INFORMATION
    Var Ret As Long

    If Api_FileTimeToSystemTime(ft, st) Then
        Ret = Api_GetTimeZoneInformation(tz)
        Ret = Api_SystemTimeToTzSpecificLocalTime(tz, st, lt)

        buff = Right$(Str$(10000 + lt.wYear), 4) & "年" & Right$(Str$(100 + lt.wMonth), 2) &
"月" & Right$(Str$(100 + lt.wDay), 2) & "日"

        If bIncludeTime Then
            buff = buff & " " & Right$(Str$(100 + lt.wHour), 2) & "時" & Right$(Str$(100 +
lt.wMinute), 2) & "分" & Right$(Str$(100 + lt.wSecond), 2) & "秒"
        End If
```

```
        GetFileToSystemDate = buff
    Else
        GetFileToSystemDate = ""
    End If
End Function

'=====================================
'=
'=====================================
Declare Function GetLastSystemShutdown(bIncludeTime As Integer) As String
Function GetLastSystemShutdown(bIncludeTime As Integer) As String
    Var hKey As Long
    Var sKey As String
    Var sValueName As String
    Var ft As FILETIME
    Var cbData As Long
    Var Ret As Long

    sKey = "System¥CurrentControlSet¥Control¥Windows"
    sValueName = "ShutdownTime"

    If Api_RegOpenKeyEx(HKEY_LOCAL_MACHINE, sKey, 0, (STANDARD_RIGHTS_READ Or
KEY_QUERY_VALUE Or EY_ENUMERATE_SUB_KEYS Or KEY_NOTIFY) And (Not SYNCHRONIZE), hKey) =
ERROR_SUCCESS Then
        If hKey <> 0 Then
            cbData = Len(ft)
            If Api_RegQueryValueEx(hKey, sValueName, 0, REG_BINARY, ft, cbData) =
ERROR_SUCCESS Then
                GetLastSystemShutdown = GetFileToSystemDate(ft, bIncludeTime)
            End If
        Ret = Api_RegCloseKey(hKey)
        End If
    End If
End Function

'=====================================
'=
'=====================================
Declare Function TrimNull(startstr As String) As String
Function TrimNull(startstr As String) As String
   TrimNull = Left$(startstr, Api_lstrlenW(StrAdr(startstr)))
End Function

'=====================================
'=
'=====================================
Declare Function GetLocalComputerName() As String
Function GetLocalComputerName() As String
    Var tmp As String

    tmp = Space$(MAX_COMPUTERNAME)

    If Api_GetComputerName(tmp, Len(tmp)) <> 0 Then
        GetLocalComputerName = TrimNull(tmp)
    End If
End Function

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
   Text1.SetWindowText GetLocalComputerName() & " の最終シャットダウン日時"
   Edit1.SetWindowText ""
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
```

```
Sub Button1_on()
   Var buff As String
   Var bIncludeTime As Integer

   bIncludeTime = Check1.GetCheck
   buff = GetLastSystemShutdown(bIncludeTime)
   Edit1.SetWindowText buff
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

最小化されているウィンドウを復元する

例では、可視なしに設定したフォームを最小化し、3秒後に元の大きさに復元しています。
IsIconic ウィンドウが最小化されているかどうかを判断
OpenIcon 最小化されているウィンドウを、元のサイズに戻す
Sleep 指定した時間の間、処理を停止




```
'=====================================
'= 最小化されているウィンドウを復元する
'=    (OpenIcon.bas)
'=====================================
#include "Windows.bi"

' ウィンドウが最小化されているかどうかを判断
Declare Function Api_IsIconic& Lib "user32" Alias "IsIconic" (ByVal hWnd&)

' 最小化されているウィンドウを、元のサイズに戻す
Declare Function Api_OpenIcon& Lib "user32" Alias "OpenIcon" (ByVal hWnd&)

' 指定した時間の間、処理を停止
Declare Sub Api_Sleep Lib "kernel32" Alias "Sleep" (ByVal dwMilliseconds&)

Var Shared Text1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var Ret As Long

    MinimizeWindow
    ShowWindow -1
    CallEvent

    '3秒待つ(単位:ミリ秒)
    Api_Sleep 3000
```

```
    'ウィンドウが最小化されている場合、復元しアクティブに
    If Api_IsIconic(GethWnd) <> False Then
        Ret = Api_OpenIcon(GethWnd)
        Text1.SetWindowText "最小化から復元しました！"
    End If
End Sub

'========================================
'=
'========================================
While 1
    WaitEvent
Wend
Stop
End
```

最小化されているウィンドウの数を取得

最小化（アイコン化）されたウィンドウの数を取得します。
GetDesktopWindow Windowsのデスクトップウィンドウを識別
ArrangeIconicWindows 指定された親ウィンドウが持つ最小化（アイコン化）された子ウィンドウをすべて整列
自身を除く6個がアイコン化されている。

```
'============================================
'= 最小化されているウィンドウの数を取得
'=    (ArrangeIconicWindows.bas)
'============================================
#include "Windows.bi"

' Windowsのデスクトップウィンドウを識別。返されるポインタは、一時的なポインタ。後で使用するために保存しておくことはできない
Declare Function Api_GetDesktopWindow& Lib "user32" Alias "GetDesktopWindow" ()

' 指定された親ウィンドウが持つ最小化（アイコン化）された子ウィンドウをすべて整列
Declare Function Api_ArrangeIconicWindows& Lib "user32" Alias "ArrangeIconicWindows"
(ByVal hWnd&)
Var Shared Text1 As Object
Var Shared Button1 As Object
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'============================================
'=
'============================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hWnd As Long
    Var Ret As Long

    hWnd = Api_GetDesktopWindow()
    Ret = Api_ArrangeIconicWindows(ByVal hWnd)
    Text1.SetWindowText "アイコン化された数:" & Str$(Ret)
End Sub

'============================================
'=
'============================================
While 1
    WaitEvent
```

```
Wend
Stop
End
```

サイズグリップを作成（Ⅰ）

サイズグリップを（擬似的に）作成してみます。
ReleaseCapture マウスのキャプチャを解放
SendMessage ウィンドウにメッセージを送信
GetSystemMetrics 表示要素の寸法とシステム構成の設定を取得

フォント名「Marlett」で小文字の「o」を書き込んだテキストボックス（境界線なし）を、常時フォームの右下に固定しています。

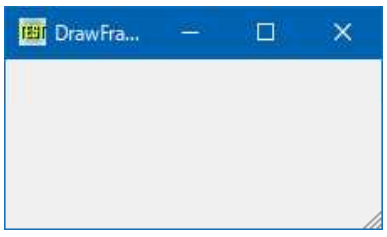
参考
Marlett について
http://www.itmedia.co.jp/help/tips/windows/w0288.html

```
'=======================================
'= サイズグリップを（擬似的に）作成
'=    (SimulateSizeGrip.bas)
'=======================================
#include "Windows.bi"

' マウスのキャプチャを解放
Declare Function Api_ReleaseCapture& Lib "user32" Alias "ReleaseCapture" ()

' ウィンドウにメッセージを送信
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

' 表示要素の寸法とシステム構成の設定を取得
Declare Function Api_GetSystemMetrics& Lib "user32" Alias "GetSystemMetrics" (ByVal
nIndex&)

#define WM_NCLBUTTONDOWN &HA1                  '非クライアント領域で左マウスボタンを押す
#define HTBOTTOMRIGHT 17                       'ウィンドウ境界の右下隅

#define SM_CXFRAME 32                          'サイズ可変ウィンドウの境界線の、X方向の幅
#define SM_CYFRAME 33                          '  〃 Y方向の幅
#define SM_CYSIZE 31                           'タイトルバー内のビットマップの高さ
#define SM_CYBORDER 6                          'サイズ固定ウィンドウの境界線のY方向の幅

Var Shared Text1 As Object

Var Shared zX As Integer
Var Shared zY As Integer

'=======================================
'=
'=======================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var fs As Integer

    fs = 14
    Text1.Attach GetDlgItem("Text1")
    Text1.SetFontSize fs
    Text1.SetWindowSize fs, fs
    Text1.SetFontName "Marlett"
```

```
    Text1.SetWindowText "o"

    ShowWindow -1

    zX = Api_GetSystemMetrics(SM_CXFRAME) * 2
    zY = Api_GetSystemMetrics(SM_CYFRAME) * 2 + Api_GetSystemMetrics(SM_CYBORDER) +
Api_GetSystemMetrics(SM_CYSIZE)
End Sub

'=======================================
'=
'=======================================
Declare Sub Mainform_MouseDown edecl (ByVal Button%, ByVal Shift%, ByVal SX!, ByVal SY!)
Sub Mainform_MouseDown(ByVal Button%, ByVal Shift%, ByVal SX!, ByVal SY!)
    Var Ret As Long

    If Button% = 1 Then
        Ret = Api_ReleaseCapture()
        Ret = Api_SendMessage(Text1.GethWnd, WM_NCLBUTTONDOWN, HTBOTTOMRIGHT, 0)
    End If
End Sub

'=======================================
'=
'=======================================
Declare Sub MainForm_Resize edecl ()
Sub MainForm_Resize()

    Text1.MoveWindow (Getwidth - Text1.GetWidth) - zX, GetHeight - Text1.GetHeight - zY
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

サイズグリップの作成（II）

DrawFrameControlでサイズグリップを作成します。
DrawFrameControl フレームコントロールを描画
GetSystemMetrics 表示要素の寸法とシステム構成の設定を取得
GetSysColor システムの背景色を取得
GetDC デバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストの解放



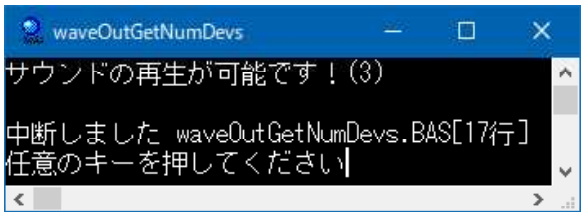

```
'=======================================
'= サイズグリップを作成
'=    (SizeGrip.bas)
'=======================================
#include "Windows.bi"

Type RECT
    Left        As Long
    Top         As Long
```

```
    Right      As Long
    Bottom     As Long
End Type

#define DFC_SCROLL 3                          'スクロールバーを描画
#define DFCS_SCROLLSIZEGRIP 8                 'ウィンドウの右下隅にあるサイズ変更グリップ

#define WM_NCLBUTTONDOWN &HA1                 '非クライアント領域で左マウスボタンを押す
#define HTBOTTOMRIGHT 17                      'ウィンドウ境界の右下隅
#define COLOR_BTNFACE 15                      'コマンドボタンの表面色

#define SM_CXFRAME 32                         'サイズ可変ウィンドウの境界線の、X方向の幅
#define SM_CYFRAME 33                         'サイズ可変ウィンドウの境界線の、Y方向の幅
#define SM_CYSIZE 31                          'タイトルバー内のビットマップの高さ
#define SM_CYBORDER 6                         'サイズ固定ウィンドウの境界線のY方向の幅

' 指定されたタイプとスタイルを備える、ボタンやスクロールバーなどのフレームコントロールを描画
Declare Function Api_DrawFrameControl& Lib "user32" Alias "DrawFrameControl" (ByVal
hDC&, lpRect As RECT, ByVal un1&, ByVal un2&)

' 表示要素の寸法とシステム構成の設定を取得
Declare Function Api_GetSystemMetrics& Lib "user32" Alias "GetSystemMetrics" (ByVal
nIndex&)

' システムの背景色を取得
Declare Function Api_GetSysColor& Lib "user32" Alias "GetSysColor" (ByVal nIndex&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

Var Shared zX As Integer
Var Shared zY As Integer

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var rgbColor As Long
    rgbColor = Api_GetSysColor(COLOR_BTNFACE)     'Buttonの表面色を取得(EDE9EC)
    SetBackColor rgbColor                         'Mainformを取得色で塗りつぶす

    ShowWindow -1
    zX = Api_GetSystemMetrics(SM_CXFRAME) * 2
    zY = Api_GetSystemMetrics(SM_CYFRAME) * 2 + Api_GetSystemMetrics(SM_CYBORDER) +
Api_GetSystemMetrics(SM_CYSIZE)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Resize edecl ()
Sub MainForm_Resize()
    Var rct As RECT
    Var hDC As Long
    Var Ret As Long

    hDC = Api_GetDC(GethWnd)

    rct.Left = GetWidth - (zX + GetFontSize)
    rct.Top = GetHeight - (zY + GetFontSize)
    rct.Right = GetWidth - zX
    rct.Bottom = GetHeight - zY

    Cls

    Ret = Api_DrawFrameControl(hDC, rct, DFC_SCROLL, DFCS_SCROLLSIZEGRIP)
```

```
    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

サウンド再生が可能かどうかを調べる

waveOutGetNumDevs システムに存在するWave出力デバイスの数を取得




```
'=====================================
'= サウンド再生が可能かどうかを調べる
'=    (waveOutGetNumDevs.bas)
'=====================================

' システムに存在するWave出力デバイスの数を取得
Declare Function Api_waveOutGetNumDevs& Lib "winmm" Alias "waveOutGetNumDevs" ()

Var Ret As Long
Ret = Api_waveOutGetNumDevs()

If Ret > 0 Then
    Print "サウンドの再生が可能です！(" & Trim$(Str$(Ret)) & ")"
Else
    Print "サウンドの再生は不可能です！"
End If

Stop
End
```

サウンドの再生(Ⅰ)

サウンドを再生します。
PlaySound WAVファイルを再生する

例では、レジストリエントリー内のサウンドを再生します。

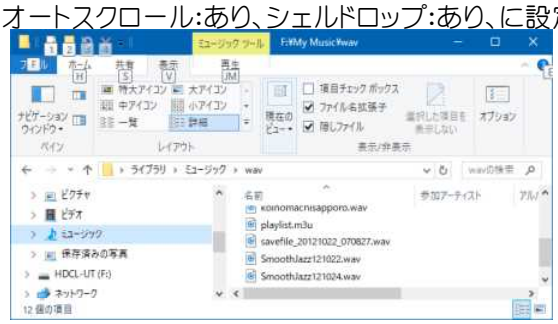

```
'=======================================
'= サウンドの再生
'=    (PlaySound.bas)
'=======================================
#include "Windows.bi"

' WAVファイルを再生する
Declare Function Api_PlaySound& Lib "winmm" Alias "PlaySoundA" (ByVal lpszName$, ByVal
hModule&, ByVal dwFlags&)

#define SND_ASYNC &H1                    '非同期的にサウンドを再生(再生の開始直後に制御を返す)
#define SND_LOOP &H8                     'lpszSoundNameパラメータをNullに設定して
                                         sndPlaySoundを再び呼び出すまでサウンドを繰返し再生
#define SND_MEMORY &H4                   'lpszSoundNameで指定したパラメータがサウンドのメモリ
                                         内データイメージを指すポインタであることを示す
#define SND_NODEFAULT &H2                'サウンドが見つからないとき、関数はデフォルトのサウンドを
                                         再生せずに制御を返す
#define SND_NOSTOP &H10                  'サウンドが現在再生中ならば、関数は要求されたサウンドを
                                         再生せず、すぐにFalseを返す
#define SND_SYNC &H0                     '同期的にサウンドを再生(再生が終了するまで制御を返さ
                                         ない)

Var Shared Combo1 As Object
Var Shared Button1 As Object

Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
Button1.Attach GetDLgItem("Button1") : Button1.SetFontSize 14

'=======================================
'=
'=======================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Combo1.AddString ".Default"
    Combo1.AddString "AppGPFault"
    Combo1.AddString "Close"
    Combo1.AddString "Maximize"
    Combo1.AddString "MenuCommand"
    Combo1.AddString "MenuPopup.Current"
    Combo1.AddString "Minimize"
    Combo1.AddString "Open"
    Combo1.AddString "RestoreDown"
    Combo1.AddString "RestoreUp"
    Combo1.AddString "SystemAsterisk"
    Combo1.AddString "SystemExclamation"
    Combo1.AddString "SystemExit"
    Combo1.AddString "SystemHand"
    Combo1.AddString "SystemQuestion"
    Combo1.AddString "SystemStart"
End Sub

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var RegEntry As String
    Var fuSound As Long
    Var Ret As Long
```

```
    'レジストリエントリを設定
    RegEntry = Combo1.GetText(Combo1.GetCursel)

    '再生方法を設定
    fuSound = SND_ASYNC Or SND_NODEFAULT

    'ウェーブフォームサウンドを再生
    Ret = Api_PlaySound(RegEntry, 0, fuSound)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

サウンドの再生（II）

WAVファイルをドラッグ＆ドロップし再生します。
PlaySound WAVファイルを再生
GetShortPathName ファイルの短い型式のパス名を取得

ファイルをドラッグ&ドロップしてEditBoxに入れ再生します。
Edit1は、複数行入力:あり、垂直オートスクロール:あり、シェルドロップ:あり、に設定しておきます。



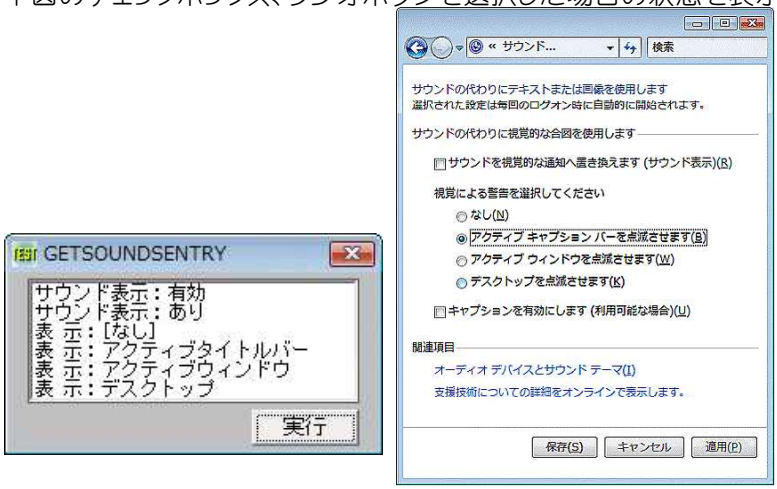
```
'=====================================
'= WAVファイルの再生
'=    (PlaySound2.bas)
'=====================================
#include "Windows.bi"

' WAVファイルを再生する
Declare Function Api_PlaySound& Lib "winmm" Alias "PlaySoundA" (ByVal lpszName$, ByVal
hModule&, ByVal dwFlags&)

' ファイルの短い形式のパス名を取得
Declare Function Api_GetShortPathName& Lib "Kernel32" Alias "GetShortPathNameA" (ByVal
lpszLongPath$, ByVal lpszShortPath$, ByVal lBuffer&)

#define SND_Alias &H10000                    'pszSoundはシステムイベントのエイリアス
#define SND_Alias_ID &H110000                'pszSoundは定義済みサウンド識別子
#define SND_APPLICATION &H80                 'アプリケーション特有の関連づけで再生
#define SND_ASYNC &H1                        '非同期再生。再生が始まるとすぐに制御を返す
#define SND_FILENAME &H20000                 'pszSoundはファイル名
#define SND_LOOP &H8                         '繰り返し再生
#define SND_MEMORY &H4                       'サウンドイベントのファイルはメモリにロード
#define SND_NODEFAULT &H2                    '指定のサウンドが見つからない時、警告音などを出さずに
終了

#define SND_NOSTOP &H10                      'すでに他のサウンドが再生されている時、指定のサウンドを
再生しない

#define SND_NOWait &H2000                    'ドライバがビジーの時はすぐに制御を返す
#define SND_PURGE &H40                       'サウンドの再生を停止
#define SND_RESOURCE &H40004                 'pszSoundはリソース識別子を表わす
#define SND_SYNC &H0                         '同期再生。再生が終了するまで制御を返さない
```

```
Var Shared Edit1 As Object
Var Shared Button1 As Object
Var Shared Button2 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14

Var Shared PathName As String
Var Shared Sound As String

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    PathName = Edit1.GetWindowText
    Sound = Space$(260)

    Ret = Api_GetShortPathName(PathName, Sound, Len(Sound))

    If Len(Sound) > 0 Then
        Ret = Api_PlaySound(Sound, 0, SND_ASYNC Or SND_NODEFAULT Or SND_FILENAME)

        If Ret <> 0 Then
            SetWindowText "Soundは再生中！"
        Else
            SetWindowText "Soundは再生できません！"
        End If
    End If
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Ret As Long

    Ret = Api_PlaySound(ByVal 0, 0, 0)

    SetWindowText "停止しました！"
End Sub

'=====================================
'= シェルドロップされたファイル名を取得
'=====================================
Declare Sub Edit1_DropFiles edecl (ByVal DF As Long)
Sub Edit1_DropFiles(ByVal DF As Long)
    Var CN As Long
    CN = GetDropFileCount(DF)
    PathName = GetDropFileName(DF, 0)
    Edit1.SetWindowText PathName
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

サウンド表示情報を取得

SystemParametersInfo システム全体に関するパラメータを取得・設定
lstrcpy 文字列をコピーする

下図のチェックボックス、ラジオボタンを選択した場合の状態を表示させています。




```
'=======================================
'= サウンド表示情報を取得
'=    (SoundSentry.bas)
'=======================================
#include "Windows.bi"

Type tagSOUNDSENTRY
    cbSize                  As Long
    dwFlags                 As Long
    iFSTextEffect           As Long
    iFSTextEffectMSec       As Long
    iFSTextEffectColorBits  As Long
    iFSGrafEffect           As Long
    iFSGrafEffectMSec       As Long
    iFSGrafEffectColor      As Long
    iWindowsEffect          As Long
    iWindowsEffectMSec      As Long
    lpszWindowsEffectDLL    As Long
    iWindowsEffectOrdinal   As Long
End Type

#define SSF_AVAILABLE &H2                   'サウンド機能使用可能
#define SSF_INDICATOR &H4                   'SoundSentryがついている間、システムアイコンを表示
#define SSF_SOUNDSENTRYON &H1               'サウンド機能ON
#define SPI_GETSOUNDSENTRY 64               'ユーザー補助機能のサウンド解説に関する情報を定義する
                                            SOUNDSENTRY構造体を取得

#define MAX_PATH 260
#define SSWF_CUSTOM 4                       'lpszWindowsEffectDLLで指定するDLLからエクスポー
                                            トされるSoundSentryProcを呼び出す(Windows9x)
#define SSWF_DISPLAY 3                      'ディスプレイ全体を点滅
#define SSWF_NONE 0                         'ビジュアル効果なし
#define SSWF_TITLE 1                        'アクティブウィンドウのキャプションバーを点滅

' システム全体に関するパラメータを取得・設定
Declare Function Api_SystemParametersInfo& Lib "user32" Alias "SystemParametersInfoA"
(ByVal uiAction&, ByVal uiParam&, pvParam As Any, ByVal fWinIni&)

' 文字列をコピーする
Declare Function Api_lstrcpy$ Lib "Kernel32" Alias "lstrcpy" (lpszString1$, lpszString2
As Any)

Var Shared List1 As Object
Var Shared Button1 As Object

List1.Attach GetDlgItem("List1") : List1.SetFontSize 12
```

```
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var tss As tagSOUNDSENTRY
    Var edll As String * MAX_PATH
    Var Ret As Long

    'リストボックスをクリア
    List1.ResetContent

    '構造体を初期化
    tss.cbSize = Len(tss)

    'サウンド表示の情報を取得
    Ret = Api_SystemParametersInfo(SPI_GETSOUNDSENTRY, Len(tss), tss, 0)

    'サウンド表示の情報を表示
    If (tss.dwFlags And SSF_SOUNDSENTRYON) = SSF_SOUNDSENTRYON Then
        List1.AddString "サウンド表示:有効"
    End If

    If (tss.dwFlags And SSF_AVAILABLE) = SSF_AVAILABLE Then
        List1.AddString "サウンド表示:あり"
    End If

    If (tss.iWindowsEffect And SSWF_NONE) = SSWF_NONE Then
        List1.AddString "表 示:[なし]"
    End If

    If (tss.iWindowsEffect And SSWF_TITLE) = SSWF_TITLE Then
        List1.AddString "表 示:アクティブタイトルバー"
    End If

    If (tss.iWindowsEffect And SSWF_WINDOW) = SSWF_WINDOW Then
            List1.AddString "表 示:アクティブウィンドウ"
    End If

    If (tss.iWindowsEffect And SSWF_DISPLAY) = SSWF_DISPLAY Then
        List1.AddString "表 示:デスクトップ"
    End If

    If (tss.iWindowsEffect And SSWF_CUSTOM) = SSWF_CUSTOM Then
        List1.AddString "表 示:カスタム"
    End If

    'DLL名が含まれているとき
    If tss.lpszWindowsEffectDLL <> 0 Then
        edll = Api_lstrcpy(edll, tss.lpszWindowsEffectDLL)
        List1.AddString "DLL名:" & Left$(edll, InStr(edll, Chr$(0)) - 1)
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

視覚スタイル(Visual Style)を判別

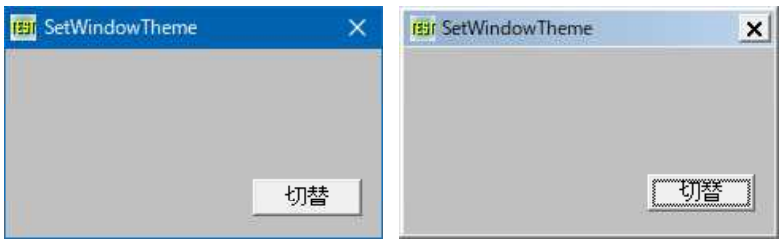
視覚スタイル(Visual Style)を判別します。
IsThemeActive ビジュアルスタイルが有効かどうかを判別

左:WindowsXPスタイル　中:マニフェストファイルを作成した場合　右:クラシックスタイル


```
'=======================================
'= 視覚スタイル(VisualStyle)を判別
'=    (IsThemeActive.bas)
'=======================================
#include "Windows.bi"

' ビジュアルスタイルが有効か判別
Declare Function Api_IsThemeActive& Lib "uxtheme" Alias "IsThemeActive" ()

Var Shared Text1 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    If Api_IsThemeActive Then
        Text1.SetWindowText "WindowsXPスタイル"
    Else
        Text1.SetWindowText "クラシックスタイル"
    End If
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

視覚スタイル(VisualStyle)を切り替える

FindWindow クラス名またはキャプションを与えてウィンドウのハンドルを取得
IsThemeActive ビジュアルスタイルが有効か判別
SetWindowTheme ウィンドウのビジュアルテーマを設定

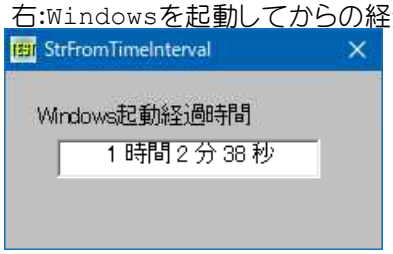


```
'=====================================
'= 視覚スタイル(VisualStyle)を切り替える
'=    (SetWindowTheme.bas)
'=====================================
#include "Windows.bi"

' クラス名またはキャプションを与えてウィンドウのハンドルを取得
Declare Function Api_FindWindow& Lib "user32" Alias "FindWindowA" (ByVal lpClassName$,
ByVal lpWindowName$)

' ビジュアルスタイルが有効か判別
Declare Function Api_IsThemeActive& Lib "uxtheme" Alias "IsThemeActive" ()

' ウィンドウのビジュアルテーマを設定
Declare Function Api_SetWindowTheme& Lib "uxtheme" Alias "SetWindowTheme" (ByVal hWnd&,
ByVal pszSubAppName&, ByVal pszSubIdList&)

Var Shared Button1 As Object

Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hWnd As Long
    Static Flag As Integer
    Var Ret As Long

    hWnd = Api_FindWindow(ByVal 0, "SetWindowTheme")

    If Api_IsThemeActive <> 0 Then
        Flag = Not Flag

        If Flag = 0 Then
            Ret = Api_SetWindowTheme(hWnd, 0, 0)
        Else
            Ret = Api_SetWindowTheme(hWnd, StrAdr("" & Chr$(0)), StrAdr("" & Chr$(0)))
        End If
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

時間を表す数値を文字列に変換

Windowsを起動してからの経過時間を取得し、文字列に変換表示します。
GetTickCount システムが起動してからの経過時間を取得
StrFromTimeInterval 時間を表す数値を文字列に変換

左:タイマーを貼り付けます　　右:Windowsを起動してからの経過時間を表示させています。



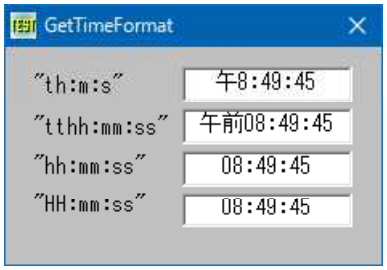
```
'=====================================
'= 時間を表す数値を文字列に変換
'=    (StrFromTimeInterval.bas)
'=====================================
#include "Windows.bi"

' システムが起動してからの経過時間を取得
Declare Function Api_GetTickCount& Lib "kernel32" Alias "GetTickCount" ()

' 時間を表す数値を文字列に変換
Declare Function Api_StrFromTimeInterval& Lib "shlwapi" Alias "StrFromTimeIntervalA"
(ByVal pszOut$, ByVal cchMax&, ByVal dwTimeMS&, ByVal dwDigits&)

Var Shared Text1 As Object
Var Shared Text2 As Object
Var Shared Timer1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Text2.Attach GetDlgItem("Text2") : Text2.SetFontSize 14
Timer1.Attach GetDlgItem("Timer1")

Var Shared numDigits As Long

'=====================================
'=
'=====================================
Declare Function FormatTimeInterval(milliseconds As Long, numDigits As Long) As String
Function FormatTimeInterval(milliseconds As Long, numDigits As Long) As String
    Var S As String * 50
    numDigits = 7

    FormatTimeInterval = Left$(S, Api_StrFromTimeInterval(S, Len(S), milliseconds,
numDigits))
End Function

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Timer1.SetInterval 10
    Timer1.Enable -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Text2.SetWindowText FormatTimeInterval(Api_GetTickCount, numDigits)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

時刻のフォーマット

SYSTEMTIMEで取得した時刻を地域の時刻にフォーマットします。
GetThreadLocale スレッドのロケールIDを取得
SetThreadLocale スレッドのロケールIDを設定
GetLocalTime ローカルタイムを取得

GetTimeFormat 時刻をフォーマットし、指定された地域に対応する時刻文字列を作成

```
Ret = Api_GetTimeFormat(0, 0, ST, "hh:mm:ss JST", fTime, Len(fTime))
        h       12時間制。先頭に「0」が付かない
        hh      12時間制。必要に応じ先頭に「0」が付く
        H       24時間制。先頭に「0」が付かない
        HH      24時間制。必要に応じ先頭に「0」が付く
        m       分。先頭に「0」が付かない
        mm      分。必要に応じ先頭に「0」が付く
        s       秒。先頭に「0」が付かない
        ss      秒。必要に応じ先頭に「0」が付く
        t       時刻マーカー。「A」「P」
        tt      時刻マーカー。「AM」「PM」「午前」「午後」
```

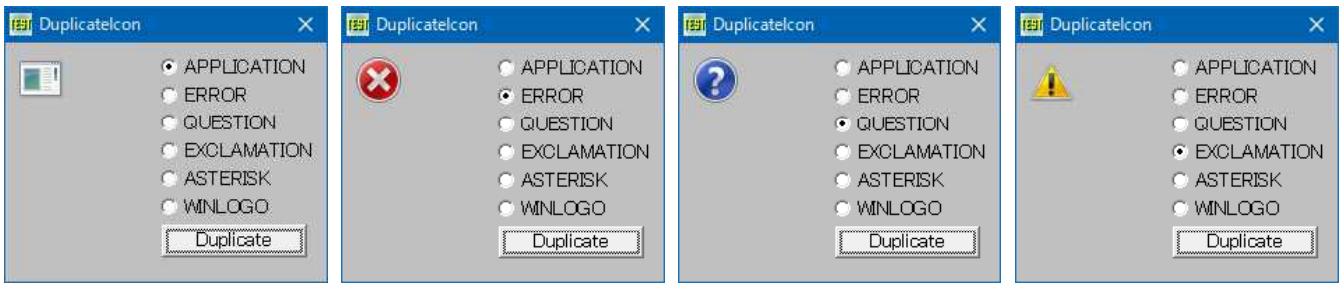



```
'=====================================
'= 時刻のフォーマット
'=    (GetTimeFormat.bas)
'=====================================
#include "Windows.bi"

#define LOCALE_SYSTEM_DEFAULT &H400          'システムのデフォルトロケール
#define LOCALE_USER_DEFAULT &H800            '現在のユーザのデフォルトロケール
#define LOCALE_NOUSEROVERRIDE -2147483648 'システムの既定の設定
#define LOCALE_USE_CP_ACP &H40000000         'ロケールのコードページの代わりに、システムのANSIコードページを使う
#define TIME_NOMINUTESORSECONDS &H1          '分と秒を使わない
#define TIME_NOSECONDS &H2                   '秒を使わない
#define TIME_NOTIMEMARKER &H4                '時刻マーカーを使わない(AM/PM)
#define TIME_FORCE24HOURFORMAT &H8           '24時間制で表記

Type SYSTEMTIME
    wYear              As Integer
    wMonth             As Integer
    wDayOfWeek         As Integer
    wDay               As Integer
    wHour              As Integer
    wMinute            As Integer
    wSecond            As Integer
    wMilliseconds      As Integer
End Type

' 時刻をフォーマットし、指定された地域に対応する時刻文字列を作成
Declare Function Api_GetTimeFormat& Lib "kernel32" Alias "GetTimeFormatA" (ByVal
Locale&, ByVal dwFlags&, lpTime As SYSTEMTIME, ByVal lpFormat As Any, ByVal lpTimeStr$,
ByVal cchTime&)

' 呼び出し側スレッドの現在のロケールを取得
Declare Function Api_GetThreadLocale& Lib "Kernel32" Alias "GetThreadLocale" ()

' 呼び出し側スレッドの現在のロケールを設定
Declare Function Api_SetThreadLocale& Lib "kernel32" Alias "SetThreadLocale" (ByVal
Locale&)

' ローカルタイムを取得
Declare Sub Api_GetLocalTime Lib "Kernel32" Alias "GetLocalTime" (lpSytemTime As
SYSTEMTIME)

Var Shared Text(7) As Object
Var Shared Timer1 As Object
```

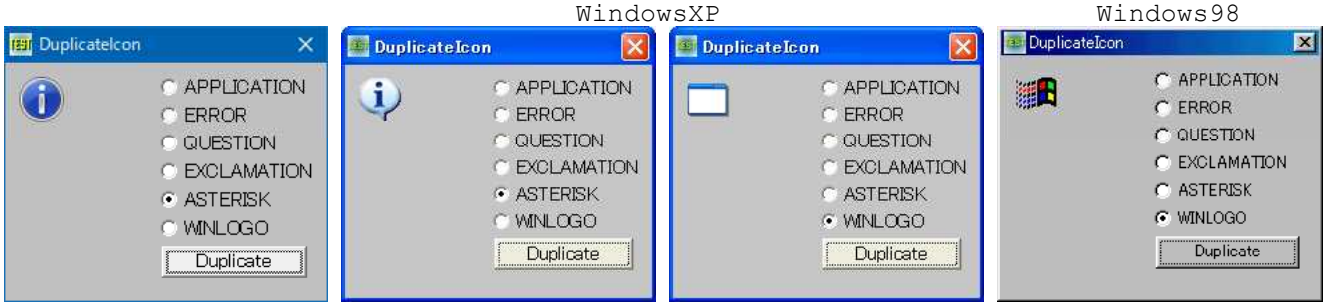
```
For i = 0 To 7
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1))) : Text(I).SetFontSize 14
Next
Timer1.Attach GetDlgItem("Timer1")

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var st As SYSTEMTIME
    Var fTime As String * 256
    Var Ret As Long

    If Api_GetThreadLocale <> LOCALE_SYSTEM_DEFAULT Then
        Ret = Api_SetThreadLocale(LOCALE_SYSTEM_DEFAULT)
    End If

    Api_GetLocalTime st

    Ret = Api_GetTimeFormat(0, 0, st, "th:m:s", fTime, Len(fTime))
    Text(4).SetWindowText Left$(fTime, Ret)

    Ret = Api_GetTimeFormat(0, 0, st, "tthh:mm:ss", fTime, Len(fTime))
    Text(5).SetWindowText Left$(fTime, Ret)

    Ret = Api_GetTimeFormat(0, 0, st, "hh:mm:ss", fTime, Len(fTime))
    Text(6).SetWindowText Left$(fTime, Ret)

    Ret = Api_GetTimeFormat(0, 0, st, "HH:mm:ss", fTime, Len(fTime))
    Text(7).SetWindowText Left$(fTime, Ret)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

システムアイコンの複製

選択したシステムアイコンを複製し、表示させています。
LoadIcon アイコンの読込
DuplicateIcon システムアイコンの複製
DrawIcon アイコンの描画



WindowsXP　　Windows98




```
'=====================================
'= システムアイコンの複製
'=     (DuplicateIcon.bas)
'=====================================
#include "Windows.bi"

#define IDI_APPLICATION 32512
#define IDI_HAND 32513
#define IDI_ERROR 32513                    'IDI_HAND
#define IDI_QUESTION 32514
#define IDI_WARNING 32515                  'IDI_EXCLAMATION
#define IDI_EXCLAMATION 32515
#define IDI_ASTERISK 32516
#define IDI_INFORMATION 32516              'IDI_ASTERISK
#define IDI_WINLOGO 32517

' アイコンの複製
Declare Function Api_DuplicateIcon& Lib "shell32" Alias "DuplicateIcon" (ByVal hInst&,
ByVal hIcon&)

' アイコンのハンドルを解放
Declare Function Api_DestroyIcon& Lib "user32" Alias "DestroyIcon" (ByVal hIcon&)

' アイコンを描画
Declare Function Api_DrawIcon& Lib "user32" Alias "DrawIcon" (ByVal hDC&, ByVal x&, ByVal
y&, ByVal exhIcon&)

' アイコン読込
Declare Function Api_LoadIcon& Lib "user32" Alias "LoadIconA" (ByVal hInstance&, ByVal
lpIconName&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

Var Shared Radio(5) As Object
Var Shared Button1 As Object

For i = 0 To 5
    Radio(i).Attach GetDlgItem("Radio" & Trim$(Str$(i + 1))) : Radio(i).SetFontSize 14
Next i
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Function Index bdecl () As Integer
Function Index()
    Index = Val(Mid$(GetDlgRadioSelect("Radio1"), 6)) - 1
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
```

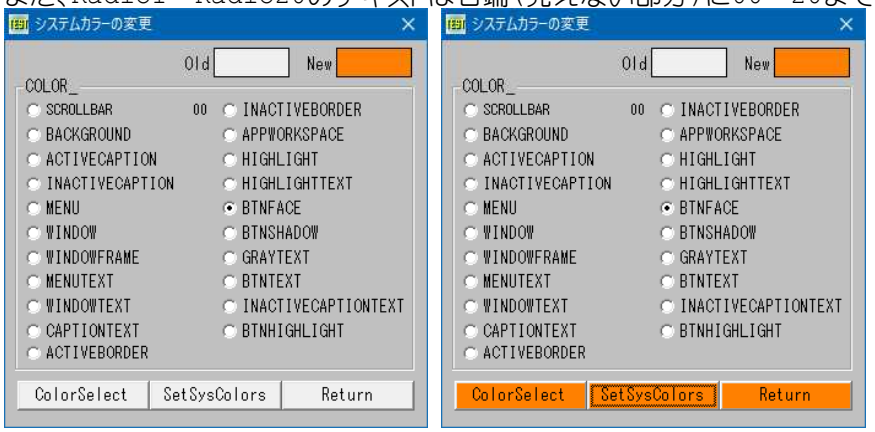
```
    Var Ret As Long
    Var hDC As Long
    Var hIcon As Long
    Var hDuplIcon As Long

    hDC = Api_GetDC(GetHwnd)

    Select Case Index
        Case 0
            hIcon = Api_LoadIcon(ByVal 0, IDI_APPLICATION)
        Case 1
            hIcon = Api_LoadIcon(ByVal 0, IDI_ERROR)
        Case 2
            hIcon = Api_LoadIcon(ByVal 0, IDI_QUESTION)
        Case 3
            hIcon = Api_LoadIcon(ByVal 0, IDI_EXCLAMATION)
        Case 4
            hIcon = Api_LoadIcon(ByVal 0, IDI_ASTERISK)
        Case 5
            hIcon = Api_LoadIcon(ByVal 0, IDI_WINLOGO)
    End Select

    Cls

    hDuplIcon = Api_DuplicateIcon(ByVal 0, hIcon)
    Ret = Api_DrawIcon(hDC, 10, 10, hDuplIcon)

    Ret = Api_DestroyIcon(hIcon)
    Ret = Api_DestroyIcon(hDuplIcon)
    Ret = Api_ReleaseDC(GetHwnd, hDC)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

システムカラーの変更

システムカラーを変更してみます。
GetSysColor システムカラーの取得
SetSysColors システムカラーの設定

変更する箇所をラジオボタンで指定すると、現在の色(Old/New)が表示されます。

色選択ボタン『ColorSelect』で色を選択すると、Newにその色が表示されます。『SetSysColors』で選択した部分のシステムカラーが変更されます。『Return』で元の色に戻ります。
リソースプロパティでラジオボタンのイベントは全てRADIOOPL_ONに書き換えておきます。
また、Radio1～Radio20のテキストは右端（見えない部分）に00～20までの文字が付加されています。




```
'=====================================
'= システムカラーの変更
'=    (SetSysColors.bas)
'=====================================
#include "Windows.bi"

' システムの背景色を取得
Declare Function Api_GetSysColor& Lib "user32" Alias "GetSysColor" (ByVal nIndex&)

' システムカラーを設定
Declare Function Api_SetSysColors& Lib "user32" Alias "SetSysColors" (ByVal nChanges&,
lpSysColor&, lpColorValues&)

#define COLOR_SCROLLBAR 0                   'スクロールバーの軸の色
#define COLOR_BACKGROUND 1                  '壁紙なしのデスクトップの色
#define COLOR_ACTIVECAPTION 2               'アクティブウィンドウのタイトルバーの色
#define COLOR_INACTIVECAPTION 3             '非アクティブウィンドウのタイトルバーのテキストの色
#define COLOR_MENU 4                        'メニューの背景色
#define COLOR_WINDOW 5                      'ウィンドウの背景色
#define COLOR_WINDOWFRAME 6                 'ウィンドウの枠の色
#define COLOR_MENUTEXT 7                    'メニュー内のテキストの色
#define COLOR_WINDOWTEXT 8                  'ウィンドウ内のテキストの色
#define COLOR_CAPTIONTEXT 9                 'アクティブウィンドウのタイトルバーのテキストの色
#define COLOR_ACTIVEBORDER 10               'アクティブウィンドウの境界の色
#define COLOR_INACTIVEBORDER 11             '非アクティブウィンドウの境界色
#define COLOR_APPWORKSPACE 12               'MDIアプリケーションの背景色
#define COLOR_HIGHLIGHT 13                  'コントロール内における選択された項目の色
#define COLOR_HIGHLIGHTTEXT 14              'コントロール内における選択された項目のテキストの色
#define COLOR_BTNFACE 15                    '3Dオブジェクトの表面色
#define COLOR_BTNSHADOW 16                  '3Dオブジェクトの影の色
#define COLOR_GRAYTEXT 17                   '淡色状態（無効状態）のテキストの色
#define COLOR_BTNTEXT 18                    'プッシュボタンのテキストの色
#define COLOR_INACTIVECAPTIONTEXT 19        '非アクティブウィンドウのテキストの色
#define COLOR_BTNHIGHLIGHT 20               '3Dオブジェクトの最も明るい色

Var Shared Part As Long
Var Shared col As byte
Var Shared oldColor As Long
Var Shared rgbColor As Long

Var Shared Picture1 As Object
Var Shared Picture2 As Object
Picture1.Attach GetDlgItem("Picture1")
Picture2.Attach GetDlgItem("Picture2")

'=====================================
'=
'=====================================
Declare Sub Color_Set edecl ()
Sub Color_Set()
```

```
    Picture1.SetFillColor oldColor
    Picture1.Line(0, 0) - (Picture1.GetWidth - 1, Picture1.Getheight - 1), Pset,, Bf
    Picture2.SetFillColor rgbColor
    Picture2.Line(0, 0) - (Picture2.GetWidth - 1, Picture2.Getheight - 1), Pset,, Bf
End Sub

'=====================================
'=
'=====================================
Declare Sub Mainform_Start edecl ()
Sub Mainform_Start()
    col = 0
    oldColor = Api_GetSysColor(COLOR_SCROLLBAR)
    rgbColor = Api_GetSysColor(COLOR_SCROLLBAR)
    Color_Set
End Sub

'=====================================
'=
'=====================================
Declare Sub RadioOpl_on edecl ()
Sub RadioOpl_on()
    Var Radio As Object
    Radio.Attach GetFocus
    static Opl$

    col = Val(Right$(Radio.GetWindowText, 2))
    Select Case col
        Case 0
            Part = COLOR_SCROLLBAR
        Case 1
            Part = COLOR_BACKGROUND
        Case 2
            Part = COLOR_ACTIVECAPTION
        Case 3
            Part = COLOR_INACTIVECAPTION
        Case 4
            Part = COLOR_MENU
        Case 5
            Part = COLOR_WINDOW
        Case 6
            Part = COLOR_WINDOWFRAME
        Case 7
            Part = COLOR_MENUTEXT
        Case 8
            Part = COLOR_WINDOWTEXT
        Case 9
            Part = COLOR_CAPTIONTEXT
        Case 10
            Part = COLOR_ACTIVEBORDER
        Case 11
            Part = COLOR_INACTIVEBORDER
        Case 12
            Part = COLOR_APPWORKSPACE
        Case 13
            Part = COLOR_HIGHLIGHT
        Case 14
            Part = COLOR_HIGHLIGHTTEXT
        Case 15
            Part = COLOR_BTNFACE
        Case 16
            Part = COLOR_BTNSHADOW
        Case 17
            Part = COLOR_GRAYTEXT
        Case 18
            Part = COLOR_BTNTEXT
        Case 19
            Part = COLOR_INACTIVECAPTIONTEXT
        Case 20
            Part = COLOR_BTNHIGHLIGHT
```

```
    End Select

    oldColor = Api_GetSysColor(Part)
    rgbColor = Api_GetSysColor(Part)
    Color_Set
End Sub

'=====================================
'=  色選択ダイアログ
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    If ChooseColor(rgbColor) = 0 Then
        rgbColor = oldColor
        Exit Sub
    End If

    Color_Set
End Sub

'=====================================
'=  システムカラーを変更する
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Ret As Long

    Ret = Api_SetSysColors(1, Part, rgbColor)
End Sub

'=====================================
'=  色を元へ戻す
'=====================================
Declare Sub Button3_on edecl ()
Sub Button3_on()
    Var Ret As Long

    Ret = Api_SetSysColors(1, Part, oldColor)
    rgbColor = oldColor
    Color_Set
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

システムカラーを取得

GetSysColor システムの背景色を取得

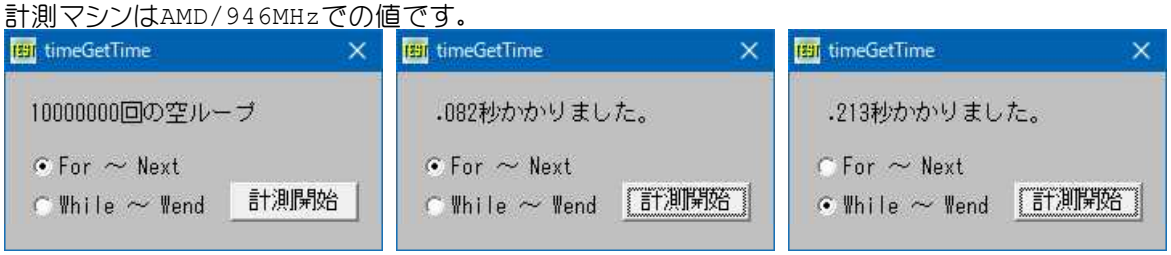
```
'=====================================
'= システムカラーを取得
'=    (GetSysColor.bas)
'=    「システムカラーの変更」参照
'=====================================
#include "Windows.bi"

' システムの背景色を取得
Declare Function Api_GetSysColor& Lib "user32" Alias "GetSysColor" (ByVal nIndex&)

#define COLOR_ACTIVEBORDER 10              'アクティブウィンドウの境界の色
#define COLOR_ACTIVECAPTION 2              'アクティブウィンドウのタイトルバーの色
#define COLOR_APPWORKSPACE 12              'MDIアプリケーションの背景色
#define COLOR_BACKGROUND 1                 '壁紙なしのデスクトップの色
#define COLOR_BTNFACE 15                   '3Dオブジェクトの表面色
#define COLOR_BTNHIGHLIGHT 20              '3Dオブジェクトの最も明るい色
#define COLOR_BTNSHADOW 16                 '3Dオブジェクトの影の色
#define COLOR_BTNTEXT 18                   'プッシュボタンのテキストの色
#define COLOR_CAPTIONTEXT 9                'アクティブウィンドウのタイトルバーのテキストの色
#define COLOR_GRAYTEXT 17                  '淡色状態(無効状態)のテキストの色
#define COLOR_HIGHLIGHT 13                 'コントロール内における選択された項目の色
#define COLOR_HIGHLIGHTTEXT 14             'コントロール内における選択された項目のテキストの色
#define COLOR_INACTIVEBORDER 11            '非アクティブウィンドウの境界色
#define COLOR_INACTIVECAPTION 3            '非アクティブウィンドウのタイトルバーのテキストの色
#define COLOR_INACTIVECAPTIONTEXT 19       '非アクティブウィンドウのテキストの色
#define COLOR_MENU 4                       'メニューの背景色
#define COLOR_MENUTEXT 7                   'メニュー内のテキストの色
#define COLOR_SCROLLBAR 0                  'スクロールバーの軸の色
#define COLOR_WINDOW 5                     'ウィンドウの背景色
#define COLOR_WINDOWFRAME 6                'ウィンドウの枠の色
#define COLOR_WINDOWTEXT 8                 'ウィンドウ内のテキストの色

Var SHared Combo1 As Object
Var Shared Button1 As Object

Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Combo1.AddString "COLOR_SCROLLBAR"
    Combo1.AddString "COLOR_BACKGROUND"
    Combo1.AddString "COLOR_ACTIVECAPTION"
    Combo1.AddString "COLOR_INACTIVECAPTION"
    Combo1.AddString "COLOR_MENU"
    Combo1.AddString "COLOR_WINDOW"
    Combo1.AddString "COLOR_WINDOWFRAME"
    Combo1.AddString "COLOR_MENUTEXT"
    Combo1.AddString "COLOR_WINDOWTEXT"
    Combo1.AddString "COLOR_CAPTIONTEXT"
    Combo1.AddString "COLOR_ACTIVEBORDER"
    Combo1.AddString "COLOR_INACTIVEBORDER"
    Combo1.AddString "COLOR_APPWORKSPACE"
    Combo1.AddString "COLOR_HIGHLIGHT"
    Combo1.AddString "COLOR_HIGHLIGHTTEXT"
    Combo1.AddString "COLOR_BTNFACE"
    Combo1.AddString "COLOR_BTNSHADOW"
    Combo1.AddString "COLOR_GRAYTEXT"
    Combo1.AddString "COLOR_BTNTEXT"
    Combo1.AddString "COLOR_INACTIVECAPTIONTEXT"
    Combo1.AddString "COLOR_BTNHIGHLIGHT"
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
```

```
Sub Button1_on()
    Var col As Integer
    Var Ret As Long

    col = Combo1.GetCursel

    Ret = Api_GetSysColor(col)

    SetBackColor Ret
    Cls
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

システム時刻をミリ秒単位で取得(Ⅰ)

システム時刻をミリ秒単位で取得します。システム時刻は Windows が起動してから経過した時間です。
timeGetTime システムが起動してからの起動時間を取得
timeBeginPeriod アプリケーションまたはデバイスドライバの最小タイマ分解能を設定
timeEndPeriod 以前にセットされた最小タイマ分解能をクリア

計算では常に、2 つの **timeGetTime**関数の戻り値の差を使います。
精度は95系では1ms、NT系では5msだそうです。
NT系の精度を上げる場合timeGetTimeの開始タイマサービスの使用直前に timeBeginPeriod を呼び出し、タイマサービスの使用終了後ただちに timeEndPeriod 関数を呼び出すそうです。
例では、for ～ next および、while ～ wend で 10000000回の空ループの時間を計測しています。
計測マシンはAMD/946MHzでの値です。



参照URL(2005/05/06現在)
http://www.microsoft.com/japan/msdn/library/default.asp?url=/japan/msdn/library/ja/jpmltimd/html/_win32_timegettime.asp

```
'=====================================
'= システム時刻をミリ秒単位で取得
'=    (timeGetTime.bas)
'=====================================
#include "Windows.bi"

' システムが起動してからの起動時間を取得
Declare Function Api_timeGetTime& Lib "winmm" Alias "timeGetTime" ()

' アプリケーションまたはデバイスドライバの最小タイマ分解能を設定
Declare Function Api_timeBeginPeriod& Lib "winmm" Alias "timeBeginPeriod" (ByVal
uPeriod&)

' 以前にセットされた最小タイマ分解能をクリア
Declare Function Api_timeEndPeriod& Lib "winmm" Alias "timeEndPeriod" (ByVal uPeriod&)

Var Shared Text1 As Object
Var Shared Radio1 As Object
Var Shared Radio2 As Object
Var Shared Button1 As Object
```

```
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Radio1.Attach GetDlgItem("Radio1") : Radio1.SetFontSize 14
Radio2.Attach GetDlgItem("Radio2") : Radio2.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Function Index bdecl () As Integer
Function Index()
    Index = Val(Mid$(GetDlgRadioSelect("Radio1"), 6)) - 1
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var i As Long
    Var Ret As Long

    Text1.SetWindowText "10000000回の空ループ"
    SetMousePointer 2

    '開始時間取得
    Ret = Api_timeBeginPeriod(1)
    nStart = Api_timeGetTime()

    '空ループ
    If Index = 0 Then
        For i = 0 To 10000000 : Next
    Else
        i = 0
        While i <= 10000000
            i = i + 1
        Wend
    End If

    ' 完了時間取得
    nEnd = Api_timeGetTime()
    Ret = Api_timeEndPeriod(1)

    ' 処理結果出力
    Text1.SetWindowText Str$((CDbl(nEnd) - CDbl(nStart)) / 1000) & "秒かかりました。"
    SetMousePointer 0
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

システム時刻をミリ秒単位で取得(II)

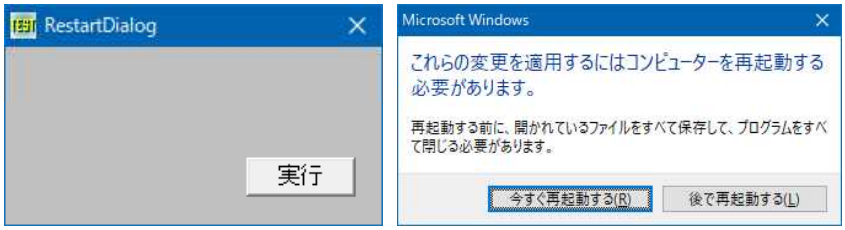
指定したキーの押下時間を測定します。timeGetTimeは、Windowsを起動してから約49日でリセットされます。
GetKeyState 指定された仮想キーコードの状態を取得
timeGetTime システムが起動してからの起動時間を取得

例では、F11キーを押下している間の時間(放したときの時刻 - 押し始め時刻)を取得しています。


```
'=====================================
'= キーの押下時間を測定
'=    (timeGetTime2.bas)
'=====================================
#include "Windows.bi"

' 指定された仮想キーコードの状態を取得
Declare Function Api_GetKeyState& Lib "User32" Alias "GetKeyState" (ByVal nVirtKey&)

' システムが起動してからの起動時間を取得
Declare Function Api_timeGetTime& Lib "winmm" Alias "timeGetTime" ()

Var Shared Text1 As Object
Var Shared Timer1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Timer1.Attach GetDlgItem("Timer1")

'=====================================
'=
'=====================================
Declare Sub Mainform_Start edecl ()
Sub Mainform_Start()
    Timer1.SetInterval 10
    Timer1.Enable -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    static timeFlag As byte
    static startTime As Long
    static EndTime As Long

    If (Api_GetKeyState(&H7A) < 0) Then 'F11
        If timeFlag = 0 Then
            startTime = Api_timeGetTime
        End If
        timeFlag = 1
    Else
        If timeFlag Then
            EndTime = Api_timeGetTime
            Text1.SetWindowText "F11押下時間:" & Str$(EndTime - startTime) & " mSec"
        End If
        timeFlag = 0
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

システム設定の変更ダイアグラムを表示

RestartDialog Windowsの終了または再起動を促すダイアログを表示



Visual Basicでは

```
Declare Function RestartDialog Lib "shell32" Alias "#59" (ByVal hParent As Long, ByVal lpPrompt As Long, ByVal dwFlags As Long) As Long

'=======================================
'= システム設定の変更ダイアログ表示
'=    (ReStartDialog.bas)
'=======================================
#include "Windows.bi"

' Windowsの終了または再起動を促すダイアログボックスを表示
Declare Function Api_RestartDialog& Lib "Shell32" Alias "RestartDialog" (ByVal hwndOwner&, ByVal lpstrReason$, ByVal uFlags&)

#define IDYES 6                                 '「はい」
#define IDNO 7                                  '「いいえ」
#define EWX_FORCE 4                             '強制
#define EWX_LOGOFF 0                            'ログオフ
#define EWX_POWEROFF 8                          'パワーオフ
#define EWX_REBOOT 2                            'リブート
#define EWX_SHUTDOWN 1                          'シャットダウン

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var temp As String
    Var Ret As Long

    temp = ""

    Ret = Api_RestartDialog(GethWnd, temp, EWX_REBOOT)

    If Ret = IDNO Then
        Cls
        Print "「いいえ」が選択されました！"
    End If
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

システム定義のサウンドを再生

ウェーブ形式のサウンドを再生します。サウンドの種類は、コントロールパネルの［サウンド］で設定できます。
MessageBeep ウェーブフォームサウンドを再生

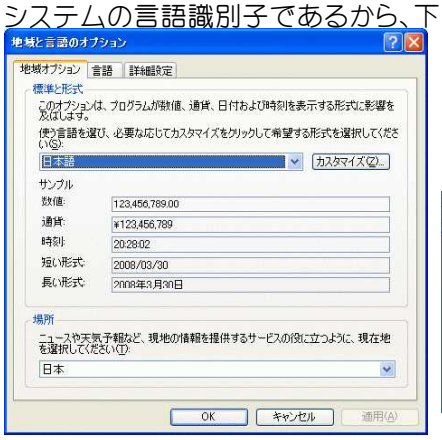

```
'=====================================
'= システム定義のサウンドを再生
'=    (MessageBeep.bas)
'=====================================
#include "Windows.bi"

#define MB_OK &H0                              '一般の警告音
#define MB_ICONHAND &H10                       'システムエラー
#define MB_ICONQUESTION &H20                   'メッセージ(問合わせ)
#define MB_ICONEXCLAMATION &H30                'メッセージ(警告)
#define MB_ICONASTERISK &H40                   'メッセージ(情報)

' ウェーブフォームサウンドを再生する
Declare Function Api_MessageBeep& Lib "user32" Alias "MessageBeep" (ByVal wType&)

Var Shared Radio(4) As Object
Var Shared Button1 As Object

For i = 0 To 4
    Radio(i).Attach GetDlgItem("Radio" & Trim$(Str$(i + 1))) : Radio(i).SetFontSize 14
Next i
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Function Index bdecl () As Integer
function Index()
    Index = Val(Mid$(GetDlgRadioSelect("Radio1"), 6)) - 1
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var WM As Long
    Var Ret As Long

    Select Case Index
        Case 0
            WM = MB_OK
        Case 1
            WM = MB_ICONHAND
        Case 2
            WM = MB_ICONQUESTION
        Case 3
            WM = MB_ICONEXCLAMATION
        Case 4
            WM = MB_ICONASTERISK
    End Select

    Ret = Api_MessageBeep(WM)
End Sub

'=====================================
'=
'=====================================
```

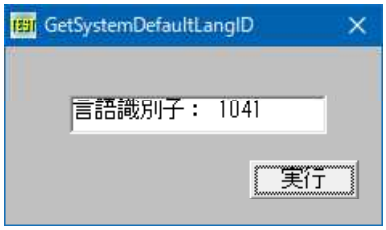
```
While 1
    WaitEvent
Wend
Stop
End
```

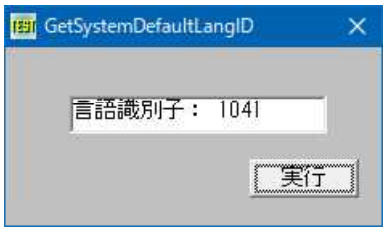
システムの既定言語識別子を取得

GetSystemDefaultLangID システムの既定言語識別子を取得

システムの言語識別子であるから、下図のように変更しても識別子は1041(日本)である。







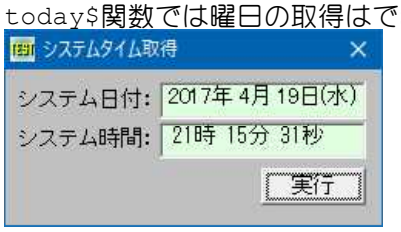

```
'=======================================
'= システムの既定言語識別子を取得
'=    (GetSystemDefaultLangID.bas)
'=======================================
#include "Windows.bi"

' システムの既定言語識別子を取得
Declare Function Api_GetSystemDefaultLangID% Lib "Kernel32" Alias
"GetSystemDefaultLangID" ()

Var Shared Text1 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    Ret = Api_GetSystemDefaultLangID()
```

```
    Text1.SetWindowText "言語識別子:" & Str$(Ret)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

システムの現在日付･時刻を取得

GetSystemTime システムの現在の日付と時刻を取得

日付･曜日･時刻(世界協定時刻)を取得します。
today$関数では曜日の取得はできない



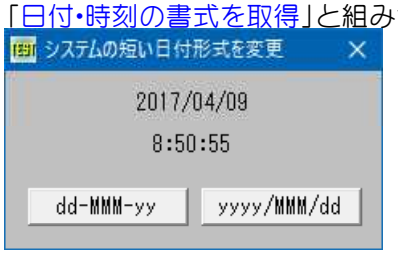

```
'=====================================
'= システムの現在日付･時刻を取得
'=    (GetSystemTime.bas)
'=====================================
#include "Windows.bi"

Type SYSTEMTIME
    wYear               As Integer
    wMonth              As Integer
    wDayOfWeek          As Integer
    wDay                As Integer
    wHour               As Integer
    wMinute             As Integer
    wSecond             As Integer
    wMilliseconds       As Integer
End Type

' 現在の世界協定時刻の取得
Declare Sub Api_GetSystemTime Lib "Kernel32" Alias "GetSystemTime" (lpSystemTime As
SYSTEMTIME)

Var Shared Text(3) As Object
Var Shared Button1 As Object

For i = 0 To 3
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1)))
    Text(i).SetFontSize 14
Next i
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var st As SYSTEMTIME

    'システムの現在の日付と時刻を取得
    Api_GetSystemTime st
```

```
    'システムの現在の日付と時刻を表示
    Text(0).SetWindowText Str$(st.wYear) & "年" & Str$(st.wMonth) & "月" & Str$(st.wDay) &
"日" & "(" & kMid$("日月火水木金土", st.wDayOfWeek + 1, 1) & ")"
    Text(1).SetWindowText Str$(st.wHour) & "時" & Str$(st.wMinute) & "分" &
Str$(st.wSecond) & "秒"

End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

システムの短い日付形式を変更

システムの地域設定の短い日付形式をプログラムによって変更します。
GetDateFormat 日付表示の書式を取得
GetTimeFormat 時刻をフォーマットし、指定された地域に対応する時刻文字列を作成
SetLocaleInfo 地域情報を設定
PostMessage メッセージをポスト
GetSystemDefaultLCID システムの既定ロケール識別子を取得

「日付・時刻の書式を取得」と組み合わせています。



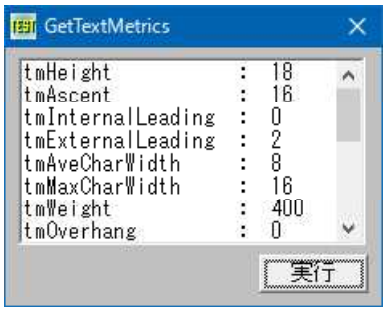

```
'=====================================
'= システムの短い日付形式を変更
'=    (SetLocaleInfo.bas)
'=====================================
#include "Windows.bi"

#define LOCALE_SSHORTDATE &H1F              '短い形式
#define WM_SETTINGCHANGE &H1A               '変更を通知
#define HWND_BROADCAST &HFFFF               'トップレベルウィンドウ に対してメッセージを送る

Type SYSTEMTIME
    wYear               As Integer
    wMonth              As Integer
    wDayOfWeek          As Integer
    wDay                As Integer
    wHour               As Integer
    wMinute             As Integer
    wSecond             As Integer
    wMilliseconds       As Integer
End Type

' 日付表示の書式を取得
Declare Function Api_GetDateFormat& Lib "Kernel32" Alias "GetDateFormatA" (ByVal
Locale&, ByVal dwFlags&, lpDate As SYSTEMTIME, ByVal lpFormat$, ByVal lpDateStr$, ByVal
cchDate&)

' 時刻をフォーマットし、指定された地域に対応する時刻文字列を作成
Declare Function Api_GetTimeFormat& Lib "kernel32" Alias "GetTimeFormatA" (ByVal
Locale&, ByVal dwFlags&, lpTime As SYSTEMTIME, ByVal lpFormat As Any, ByVal lpTimeStr$,
ByVal cchTime&)
```

```
' 地域情報を設定
Declare Function Api_SetLocaleInfo& Lib "kernel32" Alias "SetLocaleInfoA" (ByVal
Locale&, ByVal LCType&, ByVal lpLCData$)

' スレッドに関連付けられているメッセージキューにメッセージをポストする
Declare Function Api_PostMessage& Lib "user32" Alias "PostMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, ByVal lParam&)

' システムの既定ロケール識別子を取得
Declare Function Api_GetSystemDefaultLCID& Lib "kernel32" Alias "GetSystemDefaultLCID"
()

Var Shared Text(1) As Object
Var Shared Button(1) As Object
Var Shared Timer1 As Object

For i = 0 To 1
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1))) : Text(i).SetFontSize 14
    Button(i).Attach GetDlgItem("Button" & Trim$(Str$(i + 1))) : Button(i).SetFontSize 14
Next
Timer1.Attach GetDlgItem("Timer1")

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Timer1.SetInterval 100
    Timer1.Enable -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var Buffer As String
    Var st As SYSTEMTIME
    Var Ret As Long

    st.wDay = Val(Right$(Today$, 2))
    st.wMonth = Val(Mid$(Today$, 6, 2))
    st.wYear = Val(Left$(Today$, 4))

    st.wHour = Val(Left$(Time$, 2))
    st.wMinute = Val(Mid$(Time$, 4, 2))
    st.wSecond = Val(Right$(Time$, 2))

    Buffer = String$(255, 0)
    Ret = Api_GetDateFormat(ByVal 0, 0, st, ByVal 0, Buffer, Len(Buffer))
    Buffer = Left$(Buffer, InStr(1, Buffer, Chr$(0)) - 1)
    Text(0).SetWindowText Buffer

    Buffer = String$(255, 0)
    Ret = Api_GetTimeFormat(ByVal 0, 0, st, ByVal 0, Buffer, Len(Buffer))
    Buffer = Left$(Buffer, InStr(1, Buffer, Chr$(0)) - 1)
    Text(1).SetWindowText Buffer
End Sub

'=====================================
'= dd-MMM-yy
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var dwLCID As Long
    Var Ret As Long

    dwLCID = Api_GetSystemDefaultLCID()
    If Api_SetLocaleInfo(dwLCID, LOCALE_SSHORTDATE, "dd-MMM-yy") = False Then
       A% = MessageBox("", "Failed", 0, 2)
```

```
        Exit Sub
    End If

    Ret = Api_PostMessage(HWND_BROADCAST, WM_SETTINGCHANGE, 0, 0)
End Sub

'=====================================
'= yyyy/MMM/dd
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var dwLCID As Long
    Var Ret As Long

    dwLCID = Api_GetSystemDefaultLCID()
    If Api_SetLocaleInfo(dwLCID, LOCALE_SSHORTDATE, "yyyy/MMM/dd") = False Then
        A% = MessageBox("", "Failed", 0, 2)
        Exit Sub
    End If

    Ret = Api_PostMessage(HWND_BROADCAST, WM_SETTINGCHANGE, 0, 0)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

システムフォントの情報を取得

TEXTMETRIC構造体から、システムフォントの情報を取得します。
GetTextMetrics フォントに関する情報を取得
GetDesktopWindow デスクトップウィンドウのハンドルを取得
GetWindowDC ウィンドウ全体のデバイスコンテキストを取得
ReleaseDC デバイスコンテキストの解放
SetMapMode 指定のデバイスコンテキストのマッピングモードを指定



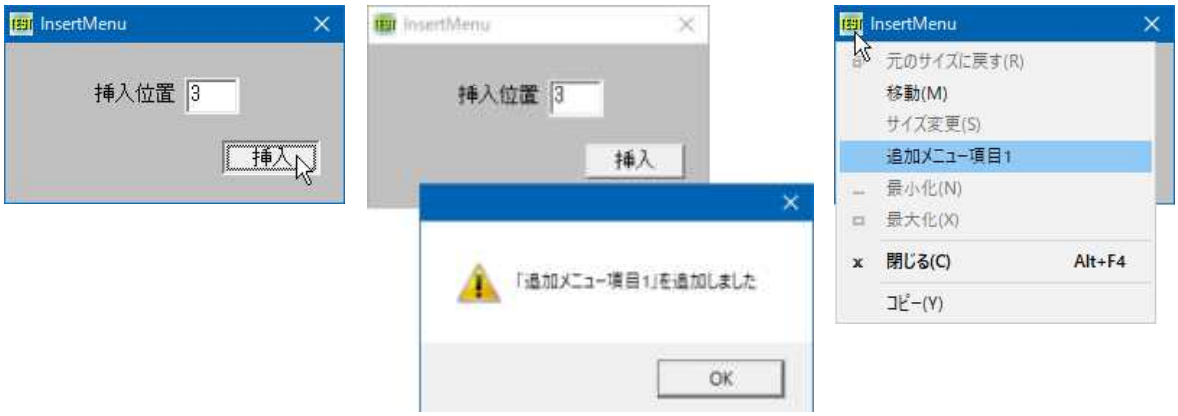

```
'=====================================
'= システムフォント情報を取得
'=    (GetTextMetrics.bas)
'=====================================
#include "Windows.bi"

Type TEXTMETRIC
    tmHeight          As Long              'フォントの高さ(tmAscent+tmDescent)
    tmAscent          As Long              'ベースラインから一番上までの高さ
    tmDescent         As Long              'ベースラインから一番下までの高さ
    tmInternalLeading As Long              'tmHeightメンバが示す高さに含まれる、上部スペースの
                                           高さ
    tmExternalLeading As Long              'フォントを描画する際の、行同士の幅
    tmAveCharWidth    As Long              'フォントの平均の幅(イタリック体では超える場合もある)
```

```
    tmMaxCharWidth    As Long              'フォントを描画する際に必要な幅(最も広い幅)
    tmWeight          As Long              'フォントの太さ(700以上がBold)
    tmOverhang        As Long              '太字やイタリック体などのフォントに付加される幅
    tmDigitizedAspectX As Long             'デバイス コンテキストの水平アスペクト値
    tmDigitizedAspectY As Long             'デバイス コンテキストの垂直アスペクト値
    tmFirstChar       As Byte              'フォントに含まれる文字中の、先頭の文字コード
    tmLastChar        As Byte              'フォントに含まれる文字中の、最後の文字コード
    tmDefaultChar     As Byte              'フォントに含まれない文字を描画するための文字コード
    tmBreakChar       As Byte              '単語と単語の間を示す文字コード
    tmItalic          As Byte              'フォントがイタリック体のときは0以外の値
    tmUnderlined      As Byt               '下線付きフォントのときは0以外の値
    tmStruckOut       As Byte              '打ち消しフォントのときは0以外の値
    tmPitchAndFamily  As Byte              '下位4ビットにフォントのピッチおよび属性
    tmCharSet         As Byte              'フォントの文字セット
End Type

' フォントに関する情報を取得
Declare Function Api_GetTextMetrics& Lib "gdi32" Alias "GetTextMetricsA" (ByVal hDC&,
lpMetrics As TEXTMETRIC)

' Windowsのデスクトップウィンドウを識別。返されるポインタは、一時的なポインタ。後で使用するために保存しておくことはできない
Declare Function Api_GetDesktopWindow& Lib "user32" Alias "GetDesktopWindow" ()

' ウィンドウ全体のデバイスコンテキストを取得
Declare Function Api_GetWindowDC& Lib "user32" Alias "GetWindowDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

' 指定のデバイスコンテキストのマッピングモードを設定
Declare Function Api_SetMapMode& Lib "gdi32" Alias "SetMapMode" (ByVal hDC&, ByVal
nMapMode&)

Var Shared List1 As Object
Var Shared Button1 As Object

List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'====================================
'=
'====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var tm As TEXTMETRIC
    Var hDC As Long
    Var hWnd As Long
    Var PrevMapMode As Long
    Var txt As String
    Var Ret As Long

    'デスクトップのウィンドウハンドル取得
    hWnd = Api_GetDesktopWindow()

    'デスクトップのデバイスコンテキスト取得
    hDC = Api_GetWindowDC(hWnd)
    If hDC Then

        'マッピングモードをピクセルに設定
        PrevMapMode = Api_SetMapMode(hDC, MM_TEXT)

        'システムフォントのサイズを取得
        Ret = Api_GetTextMetrics(hDC, tm)

        'マッピングモードを設定
        PrevMapMode = Api_SetMapMode(hDC, PrevMapMode)

        List1.Resetcontent
        List1.AddString "tmHeight          : " & Str$(tm.tmHeight)
```

```
        List1.AddString "tmAscent           : " & Str$(tm.tmAscent)
        List1.AddString "tmInternalLeading  : " & Str$(tmInternalLeading)
        List1.AddString "tmExternalLeading  : " & Str$(tm.tmExternalLeading)
        List1.AddString "tmAveCharWidth     : " & Str$(tm.tmAveCharWidth)
        List1.AddString "tmMaxCharWidth     : " & Str$(tm.tmMaxCharWidth)
        List1.AddString "tmWeight           : " & Str$(tm.tmWeight)
        List1.AddString "tmOverhang         : " & Str$(tm.tmOverhang)
        List1.AddString "tmDigitizedAspectX : " & Str$(tm.tmDigitizedAspectX)
        List1.AddString "tmDigitizedAspectY : " & Str$(tm.tmDigitizedAspectY)
        List1.AddString "tmFirstChar        : " & Str$(tm.tmFirstChar)
        List1.AddString "tmLastChar         : " & Str$(tm.tmLastChar)
        List1.AddString "tmDefaultChar      : " & Str$(tm.tmDefaultChar)
        List1.AddString "tmBreakChar        : " & Str$(tm.tmBreakChar)
        List1.AddString "tmItalic           : " & Str$(tm.tmItalic)
        List1.AddString "tmUnderlined       : " & Str$(tm.tmUnderlined)
        List1.AddString "tmStruckOut        : " & Str$(tm.tmStruckOut)
        List1.AddString "tmPitchAndFamily   : " & Str$(tm.tmPitchAndFamily)
        List1.AddString "tmCharSet          : " & Str$(tm.tmCharSet)

        'デバイスコンテキストを解放
        Ret = Api_ReleaseDC(hWnd, hDC)
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

システムメニューに項目を挿入

システムメニューに新たな項目を挿入します。
InsertMenu 指定されたメニューに新しいメニュー項目を挿入し、他のメニュー項目を下へ移動
GetSystemMenu システムメニューのハンドル取得
GetMenuItemCount メニューの項目数を取得
GetMenuItemID メニュー項目のIDを取得




```
'=====================================
'= システムメニューに項目を挿入
'=    (InsertMenu.bas)
'=====================================
#include "Windows.bi"

' 指定されたメニューに新しいメニュー項目を挿入し、他のメニュー項目を下へ移動
Declare Function Api_InsertMenu& Lib "user32" Alias "InsertMenuA" (ByVal hMenu&, ByVal
uPosition&, ByVal uFlags&, ByVal uIDNewItem&, ByVal lpNewItem$)
```

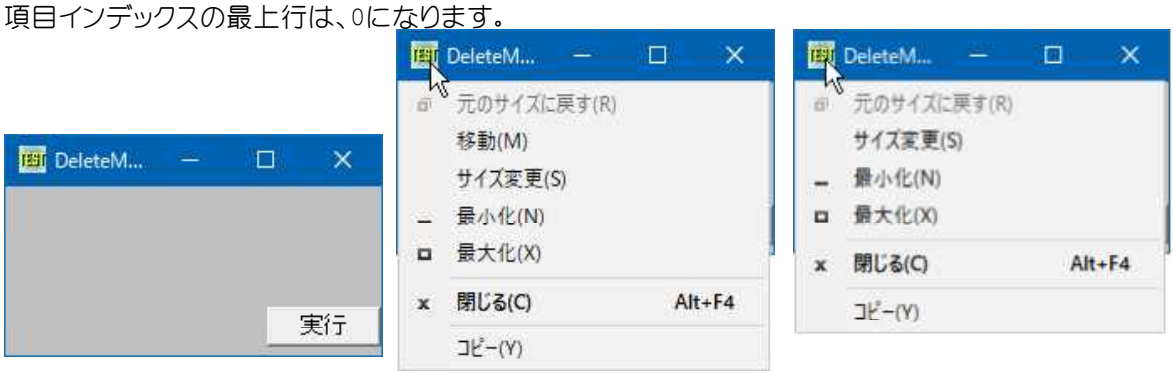
```
' システムメニューのハンドル取得
Declare Function Api_GetSystemMenu& Lib "user32" Alias "GetSystemMenu" (ByVal hWnd&,
ByVal bRevert&)

' メニューの項目数を取得
Declare Function Api_GetMenuItemCount& Lib "user32" Alias "GetMenuItemCount" (ByVal
hMenu&)

' メニュー項目のIDを取得
Declare Function Api_GetMenuItemID& Lib "user32" Alias "GetMenuItemID" (ByVal hMenu&,
ByVal nPos&)

#define SC_CLOSE &HF060                    '閉じる
#define SC_MAXIMIZE &HF030                 '最大化
#define SC_MINIMIZE &HF020                 '最小化
#define SC_MOVE &HF010                     '移動
#define SC_RESTORE &HF120                  '元のサイズに戻す
#define SC_SIZE &HF000                     'サイズ変更

#define MF_APPEND &H100                    '
#define MF_BYCOMMAND &H0                   'nPositionはメニュー項目のID
#define MF_BYPOSITION &H400                'nPositionはメニュー項目のインデックス
#define MF_CHECKED &H8                     'メニュー項目にチェックをつける
#define MF_DISABLED &H2                    'アイテムを選択不可にする
#define MF_GRAYED &H1                      'グレー表示されて選択できない
#define MF_HILITE &H80                     'メニューの項目を強調表示
#define MF_MOVE &H1000                     '
#define MF_SEPARATOR &H800                 'メニュー項目はセパレータ
#define MF_STRING &H0                      '文字列
#define MF_UNHILITE &H0                    'メニューの項目を強調表示しない

Var Shared Text1 As Object
Var Shared Edit1 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var Shared Cnt As Integer

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hMenu As Long
    Var nPos As Long
    Var mMax As Long
    Var NewStr As String
    Var Ret As Long

    Cnt = Cnt + 1
    NewStr = "追加メニュー項目" & Trim$(Str$(Cnt))

    hMenu = Api_GetSystemMenu(GethWnd, 0)  'システムメニューのハンドル取得

    mMax = Api_GetMenuItemCount(hMenu)     'メニューの項目数を取得

    nPos = Val(Edit1.GetWindowText)        '挿入場所を設定

    If nPos < 0 Or nPos > mMax Then
        Edit1.SetWindowText ""
        Edit1.SetFocus
        Exit Sub
    End If
    Ret = Api_GetMenuItemID(hMenu, nPos)

    Ret = Api_InsertMenu(hMenu, Ret, MF_BYCOMMAND, 0, NewStr)
```

```
    If  Ret <> 0 Then A% = MessageBox("", "「" & NewStr & "」を追加しました", 0, 2)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

システムメニューの削除

システムメニュー（タイトルバーのアイコンをクリックしたときに表示される）を削除します。
GetSystemMenu システムメニューのハンドル取得
DeleteMenu システムメニューの項目を削除
DrawMenuBar メニューバーを再描画

例では、実行ボタンをクリックすると、メニュー項目「移動」が削除された状態を表しています。見た目に変化はありませんが、フォームを移動させたくない場合便利です。
項目インデックスの最上行は、0になります。



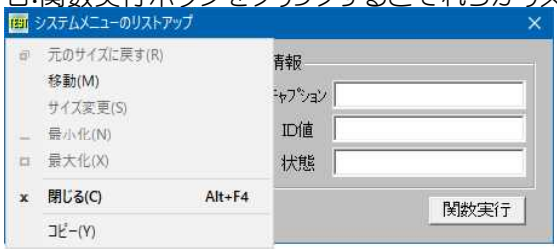

```
'=====================================
'= システムメニューの削除
'=    (DeleteMenu.bas)
'=====================================
#include "Windows.bi"

' システムメニューの項目を削除
Declare Function Api_DeleteMenu& Lib "user32" Alias "DeleteMenu" (ByVal hMenu&, ByVal
nPosition&, ByVal wFlags&)

' システムメニューのハンドル取得
Declare Function Api_GetSystemMenu& Lib "user32" Alias "GetSystemMenu" (ByVal hWnd&,
ByVal bRevert&)

' メニューバーを再描画
Declare Function Api_DrawMenuBar& Lib "user32" Alias "DrawMenuBar" (ByVal hWnd&)

#define MF_APPEND &H100                    '
#define MF_BYCOMMAND &H0                   'nPositionはメニュー項目のID
#define MF_BYPOSITION &H400                'nPositionはメニュー項目のインデックス
#define MF_CHECKED &H8                     'メニュー項目にチェックをつける
#define MF_DISABLED &H2                    '
#define MF_GRAYED &H1                      'グレー表示されて選択できない
#define MF_SEPARATOR &H800                 'メニュー項目はセパレータ
#define MF_STRING &H0                      '文字列

#define SC_CLOSE &HF060                    '閉じる
#define SC_MAXIMIZE &HF030                 '最大化
#define SC_MINIMIZE &HF020                 '最小化
#define SC_MOVE &HF010                     '移動
#define SC_RESTORE &HF120                  '元のサイズに戻す
#define SC_SIZE &HF000                     'サイズ変更
```

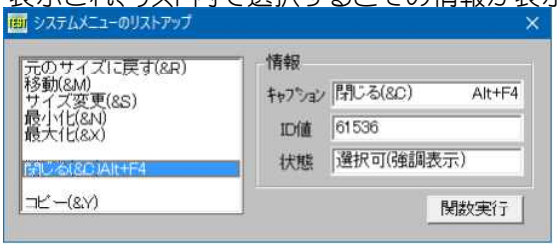
```
'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hSysMenu As Long
    Var Ret As Long

    'システムメニューのハンドルを取得
    hSysMenu = Api_GetSystemMenu(GethWnd, 0)

    'システムメニューの「移動」を削除
    Ret = Api_DeleteMenu(hSysMenu, SC_MOVE, MF_BYCOMMAND)

    'システムメニューのインデックス(最上行は0)
    'Ret = Api_DeleteMenu(hSysMenu, 1, MF_BYPOSITION)

    'メニューバーの再描画
    Ret = Api_DrawMenuBar(hSysMenu)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

システムメニューの情報を取得

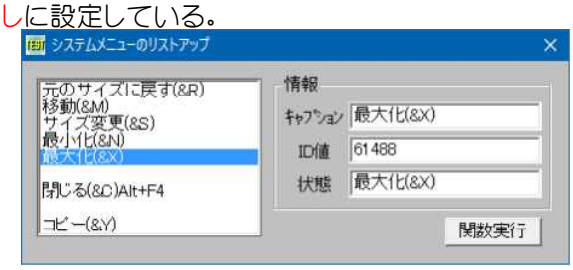
システムメニュー情報を取得します。
GetSystemMenu システムメニューハンドル取得
GetSubMenu サブメニューハンドル取得
GetMenuString メニュー文字列取得
GetMenuItemID メニュー項目ID取得
GetMenuItemCount メニュー項目数取得
GetMenuItemInfo メニュー情報取得

左:フォームのアイコンをクリックすると図のようなメニュー(システムメニュー)が表示されます。
　最大表示ボタンおよびアイコン化ボタンをありに設定しています。
右:関数実行ボタンをクリックするとそれらがリスト表示され、リスト内で選択するとその情報が表示されます。





左:最大表示ボタンおよびアイコン化ボタンをなしに設定した状態でメニュー情報を取得しています。
右:最大表示ボタンおよびアイコン化ボタンをなしに設定している。




```
'=====================================
'= システムメニューの状態取得
'=    (GetSystemMenu.bas)
'=====================================
```

```
#include "Windows.bi"

Type MENUITEMINFO
    cbSize              As Long                 '構造体のバイト数
    fMask               As Long                 '取得する情報を指定する定数の組み合わせ
    fType               As Long                 'メニュー項目のタイプを指定する定数の組み合わせ
    fState              As Long                 'メニューの状態を指定する定数の組み合わせ
    wID                 As Long                 'ユーザー定義のメニュー項目のID
    hSubMenu            As Long                 '指定のメニュー項目と関連するサブメニューのハンドル
    hbmpChecked         As Long                 'チェックマーク用のビットマップのハンドル
    hbmpUnChecked       As Long                 '未チェック時のときのビットマップハンドル
    dwItemData          As Long                 'メニュー項目と関連するユーザー定義の値
    dwTypeData          As Long                 'メニュー項目のタイプ(fMaskにMIIM_TYPEを指定したときのみ有効)
    cch                 As Long                 'メニュー項目のテキストのバイト数
End Type

' システムメニューのハンドルを取得する
Declare Function Api_GetSystemMenu& Lib "user32" Alias "GetSystemMenu" (ByVal hWnd&,
ByVal bRevert&)

' サブメニューのハンドル取得
Declare Function Api_GetSubMenu& Lib "user32" Alias "GetSubMenu" (ByVal hMenu&, ByVal
nPos&)

' メニューの文字列取得
Declare Function Api_GetMenuString& Lib "user32" Alias "GetMenuStringA" (ByVal hMenu&,
ByVal uIDItem&, ByVal lpString$, ByVal nMaxCount&, ByVal uFlag&)

' メニュー項目のID取得
Declare Function Api_GetMenuItemID& Lib "user32" Alias "GetMenuItemID" (ByVal hMenu&,
ByVal nPos&)

' メニューの項目数を取得
Declare Function Api_GetMenuItemCount& Lib "user32" Alias "GetMenuItemCount" (ByVal
hMenu&)

' メニューの情報取得
Declare Function Api_GetMenuItemInfo& Lib "user32" Alias "GetMenuItemInfoA" (ByVal
hMenu&, ByVal uItem&, ByVal fByPosition&, lpMInfo As MENUITEMINFO)

#define MF_BYCOMMAND &H0                    'uIDItemはメニュー項目のID
#define MF_BYPOSITION &H400                 'uIDItemはメニュー項目のインデックス
#define MIIM_STATE &H1                      'fStateメンバ

Var Shared List1 As Object
Var Shared Edit(2) As Object
Var Shared Text(2) As Object
Var Shared Group1 As Object
Var Shared Button1 As Object

For i = 0 To 2
    Edit(i).Attach GetDlgItem("Edit" & Trim$(Str$(i + 1))) : Edit(i).SetFontSize 14
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1))) : Text(i).SetFontSize 14
Next i
List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
Group1.Attach GetDlgItem("Group1") : Group1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hSysMenu As Long
    Var SysMenuCnt As Long
    Var Temp$ As String
    Var Ret As Long
```

```
    'リストボックスのクリア
    List1.ResetContent

    'システムメニューのハンドル取得
    hSysMenu = Api_GetSystemMenu(GetHwnd, 0)

    'システムメニューの項目数を取得
    SysMenuCnt = Api_GetMenuItemCount(hSysMenu)

    '各項目のキャプションを取得
    For i = 0 To SysMenuCnt - 1
        Temp$ = String$(256, Chr$(0))
        Ret = Api_GetMenuString(hSysMenu, i, Temp$, Len(Temp$), MF_BYPOSITION)

        ptr = InStr(Temp$, Chr$(0))
        If ptr - 1 > 0 Then
            List1.AddString Mid$(Temp$, 1, ptr - 1)
        Else
            List1.AddString ""
        End If
    Next i
End Sub

'=====================================
'=
'=====================================
Declare Sub List1_Click edecl ()
Sub List1_Click()
    Var ListIndex As Long
    Var hSysMenu As Long
    Var SysMenuCnt As Long
    Var Temp$ As String
    Var hSubMenu As Long
    Var IDItem As Long
    Var lpMInfo As MENUITEMINFO
    Var rc As Long
    Var Ret As Long

    'システムメニューのハンドル取得
    hSysMenu = Api_GetSystemMenu(GetHwnd, 0)

    '現在選択されている項目の番号を取得
    ListIndex = List1.GetCursel

    If ListIndex <> -1 Then

        'キャプション取得
        Temp$ = String$(256, Chr$(0))
        Ret = Api_GetMenuString(hSysMenu, ListIndex, Temp$, Len(Temp$), MF_BYPOSITION)

        ptr = InStr(Temp$, Chr$(0))
        If ptr - 1 > 0 Then
        Edit(0).SetWindowText Mid$(Temp$, 1, ptr - 1)
    Else
        Edit(0).SetWindowText ""
    End If

    'ID取得
    IDItem = Api_GetMenuItemID(hSysMenu, ListIndex)
    Edit(1).SetWindowText Trim$(Str$(IDItem))

    lpMInfo.cbSize = Len(lpMInfo)
    lpMInfo.fMask = MIIM_STATE

    Ret = Api_GetMenuItemInfo(hSysMenu, ListIndex, MF_BYPOSITION, lpMInfo)

    rc = lpMInfo.fState
    Select Case rc
        Case 0
            Temp$ = "選択可"
```

```
        Case 3
            Temp$ = "選択不可"
        Case 4096
            Temp$ = "選択可(強調表示)"
    End Select

    If Temp$ = "" Then Temp$ = "?????????"
        Edit(2).SetWindowText Temp$
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

システムメニューの属性変更

GetSystemMenu システムメニューのハンドル取得
GetMenuString システムメニューのラベルを取得
ModifyMenu メニューの属性を変更
DrawMenuBar メニューバーを再描画

例では、「実行」をクリック後システムメニューを見ると、「閉じる(C)」がグレー表示されているのが確認できる。
「実行」後はフォームをダブルクリックして終了させてください。

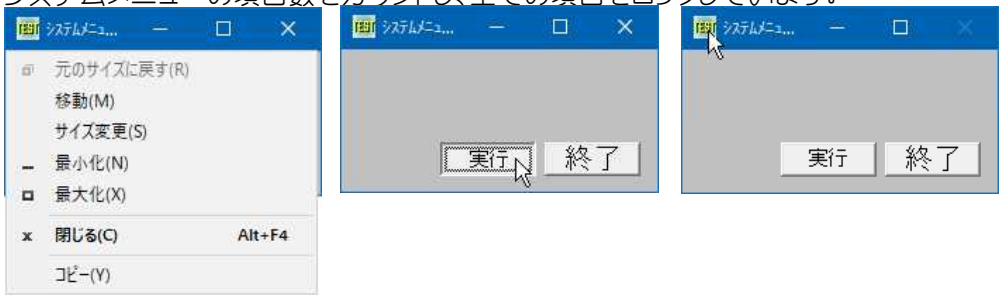

```
'=====================================
'= システムメニューの属性変更
'=    (ModifyMenu.bas)
'=====================================
#include "Windows.bi"

' システムメニューのハンドル取得
Declare Function Api_GetSystemMenu& Lib "user32" Alias "GetSystemMenu" (ByVal hWnd&,
ByVal bRevert&)

' システムメニューのラベルを取得
Declare Function Api_GetMenuString& Lib "user32" Alias "GetMenuStringA" (ByVal hMenu&,
ByVal uIDItem&, ByVal lpString$, ByVal nMaxCount&, ByVal uFlag&)

' メニューの属性を変更
Declare Function Api_ModifyMenu& Lib "user32" Alias "ModifyMenuA" (ByVal hMenu&, ByVal
nPosition&, ByVal wFlags&, ByVal wIDNewItem&, ByVal lpString$)

' メニューバーを再描画
Declare Function Api_DrawMenuBar& Lib "user32" Alias "DrawMenuBar" (ByVal hWnd&)

#define SC_CLOSE &HF060                    '閉じる
#define MF_GRAYED &H1                      'グレー表示されて選択できない
Var Shared Button1 As Object

Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
```

```
'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var MyStr As String
    Var hWnd As Long
    Var Length As Long
    Var Ret As Long

    'バッファを確保する
    MyStr = String$(250, Chr$(0))
    Length = Len(MyStr)

    'システムメニューのハンドルを確保
    hWnd = Api_GetSystemMenu(GethWnd, 0)

    'システムメニューのラベルを取得
    Ret = Api_GetMenuString(hWnd, SC_CLOSE, MyStr, Length, MF_GRAYED)

    'メニューの属性を変更
    Ret = Api_ModifyMenu(hWnd, SC_CLOSE, MF_GRAYED, 0, MyStr)

    'メニューバーを再描画
    Ret = Api_DrawMenuBar(GethWnd)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_DblClick edecl ()
Sub MainForm_DblClick()
    End
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

システムメニューをロック

GetSystemMenu システムメニューのハンドル取得
GetMenuItemCount メニューの項目数を取得
RemoveMenu システムメニューの削除
MF_BYPOSITION(&H400) nPositionはメニュー項目のインデックス

システムメニューの項目数をカウントし、全ての項目をロックしています。

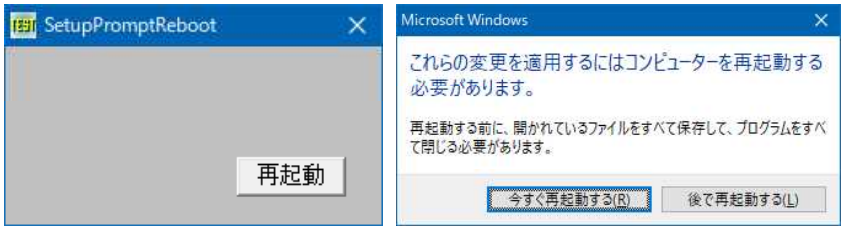



```
'=====================================
'= システムメニューをロック
'=    (GetSystemMenu2.bas)
'=====================================
```

```
#include "Windows.bi"

' システムメニューのハンドル取得
Declare Function Api_GetSystemMenu& Lib "user32" Alias "GetSystemMenu" (ByVal hWnd&,
ByVal bRevert&)

' メニューの項目数を取得
Declare Function Api_GetMenuItemCount& Lib "user32" Alias "GetMenuItemCount" (ByVal
hMenu&)

' システムメニューの削除
Declare Function Api_RemoveMenu& Lib "user32" Alias "RemoveMenu" (ByVal hMenu&, ByVal
nPosition&, ByVal wFlags&)

#define MF_BYPOSITION &H400                'nPositionはメニュー項目のインデックス

Var Shared Button1 As Object

Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hMenu As Long
    Var MenuItems As Long
    Var i As Long

    hMenu = Api_GetSystemMenu(GethWnd, 0)

    If hMenu <> 0 Then
        MenuItems = Api_GetMenuItemCount(hMenu)

        If MenuItems <> 0 Then
            For i = MenuItems - 1 To 0 Step -1
                Ret = Api_RemoveMenu(hMenu, i, MF_BYPOSITION)
            Next i
        End If
    End If
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    End
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

システムを再起動するかどうかをユーザーに尋ねる

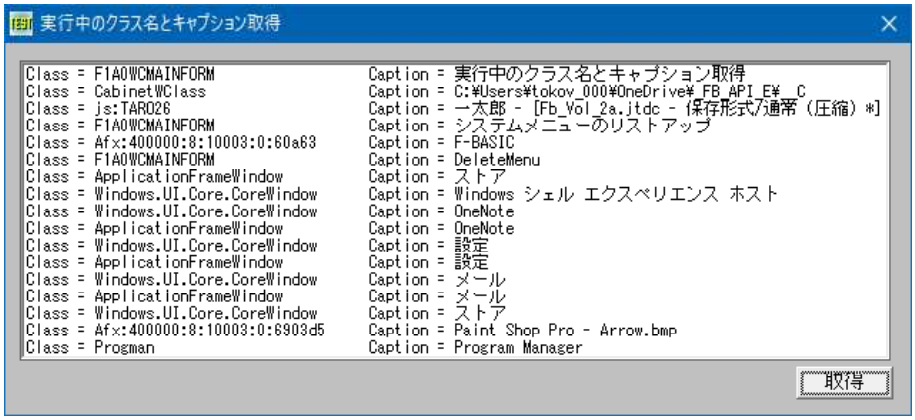
SetupPromptReboot 再起動するかどうかのダイアログボックスを表示




```
'========================================
'= システムを再起動するかどうかを尋ねる
'=    (SetupPromptReboot.bas)
'========================================
#include "Windows.bi"

'システムを再起動するかどうかをユーザーに尋ねる
Declare Function Api_SetupPromptReboot& Lib "setupapi" Alias "SetupPromptReboot" (ByRef
FileQueue&, ByVal Owner&, ByVal ScanOnly&)

'========================================
'=
'========================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    Ret = Api_SetupPromptReboot(ByVal 0, GethWnd, 0)
End Sub

'========================================
'=
'========================================
While 1
    WaitEvent
Wend
Stop
End
```

実行アプリケーションのクラス名･キャプション列挙

実行アプリケーションのクラス名とキャプションを列挙します。
GetClassName ウィンドウのクラス名を取得
GetDesktopWindow Windows のデスクトップ ウィンドウを識別
GetWindow 指定されたウィンドウと指定された関係にあるウィンドウのハンドルを取得
GetWindowLong 指定されたウィンドウに関しての情報を取得
GetWindowText ウィンドウのタイトル文字列を取得



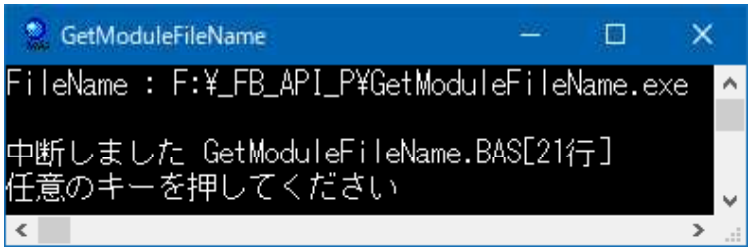

```
'========================================
'= 実行アプリケーションのクラス名･キャプション列挙
'=    (ClassNameCaption.bas)
'========================================
#include "Windows.bi"
```

```
' ウィンドウのクラス名を取得
Declare Function Api_GetClassName& Lib "user32" Alias "GetClassNameA" (ByVal hWnd&,
ByVal lpClassName$, ByVal nMaxCount&)

' Windows のデスクトップ ウィンドウを識別。返されるポインタは、一時的なポインタ。後で使用するために保存して
おくことはできない
Declare Function Api_GetDesktopWindow& Lib "user32" Alias "GetDesktopWindow" ()

' 指定されたウィンドウと指定された関係にあるウィンドウのハンドルを取得
Declare Function Api_GetWindow& Lib "user32" Alias "GetWindow" (ByVal hWnd&, ByVal wCmd&)

' 指定されたウィンドウに関しての情報を取得。また、拡張ウィンドウメモリから、指定されたオフセットにある32ビット値
を取得することもできる
Declare Function Api_GetWindowLong& Lib "user32" Alias "GetWindowLongA" (ByVal hWnd&,
ByVal nIndex&)

' ウィンドウのタイトル文字列を取得
Declare Function Api_GetWindowText& Lib "user32" Alias "GetWindowTextA" (ByVal hWnd&,
ByVal lpString$, ByVal cch&)

#define mcGWCHILD 5
#define mcGWHWNDNEXT 2
#define mcGWLSTYLE (-16)
#define mcWSVISIBLE &H10000000
#define mconMAXLEN 255

Var Shared List1 As Object
Var Button1 As Object

List1.Attach GetDlgItem("List1") : List1.SetFontSize 12
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'= クラス名取得
'=====================================
Declare Function fGetClassName(hWnd As Long) As String
Function fGetClassName(hWnd As Long) As String
    Var strBuffer As String
    Var intCount As Integer

    strBuffer = String$(mconMAXLEN - 1, 0)
    intCount = Api_GetClassName(hWnd, strBuffer, mconMAXLEN)
    If intCount > 0 Then
        fGetClassName = Left$(strBuffer, intCount)
    End If
End Function

'=====================================
'= キャプション名取得
'=====================================
Declare Function fGetCaption(hWnd As Long) As String
Function fGetCaption(hWnd As Long) As String
    Var strBuffer As String
    Var intCount As Integer

    strBuffer = String$(mconMAXLEN - 1, 0)
    intCount = APi_GetWindowText(hWnd, strBuffer, mconMAXLEN)
    If intCount > 0 Then
        fGetCaption = Left$(strBuffer, intCount)
    End If
End Function

'=====================================
'= 取得表示部
'=====================================
Declare Function fEnumWindows()
Function fEnumWindows()
    Var lx As Long
    Var lLen As Long
    Var lStyle As Long
```

```
    Var sCaption As String

    lx = Api_GetDesktopWindow()
    lx = Api_GetWindow(lx, mcGWCHILD)

        List1.ResetContent
    Do While Not lx = 0
        sCaption = fGetCaption(lx)
        If Len(sCaption) > 0 Then
            lStyle = Api_GetWindowLong(lx, mcGWLSTYLE)
            If lStyle And mcWSVISIBLE Then
                List1.AddString Left$("Class = " & fGetClassName(lx) & Space$(40), 40) &
"Caption = " & fGetCaption(lx)
            End If
        End If
        lx = Api_GetWindow(lx, mcGWHWNDNEXT)
    Loop
End Function

'=======================================
'= 取得
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    Ret = fEnumWindows
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

実行の自モジュールフルパス名を取得

GetModuleFileName ロードされている実行モジュールのフルパス名を取得



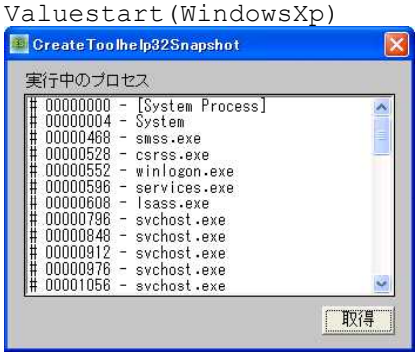

```
'=======================================
'= 実行モジュールのフルパス名を取得
'=    (GetModuleFileName.bas)
'=======================================
#include "Windows.bi"

' ロードされている実行モジュールのフルパス名を取得
Declare Function Api_GetModuleFileName& Lib "kernel32" Alias "GetModuleFileNameA" (ByVal
hModule&, ByVal lpFileName$, ByVal nSize&)

Var Buffer As String
Var FileName As String
Var Ret As Long

Buffer = string$(260, Chr$(0))
```

```
Ret = Api_GetModuleFileName(0, Buffer, len(Buffer))

FileName = left$(Buffer, instr(Buffer, Chr$(0)) - 1)

Print "FileName : ";FileName

Stop
End
```

実行中のプロセスを列挙（Ｉ）

実行中のプロセスリストを取得します。
CreateToolhelp32Snapshot プロセスのスナップショットを取得
Process32First 最初のプロセスに関する情報を取得する関数
Process32Next 2番目以降のプロセスに関する情報を取得する関数
CloseHandle オブジェクトハンドルをクローズする関数

Valuestart(WindowsXp)



Flora(Windows2000)

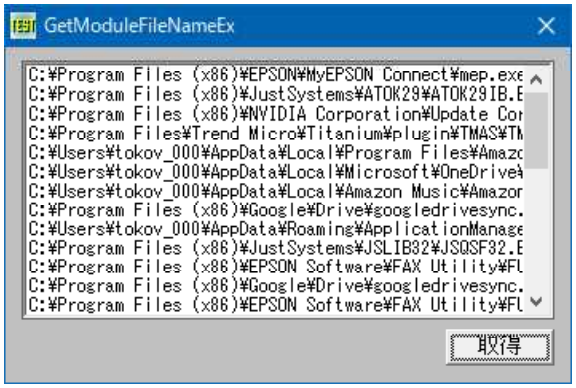
Flora(Windpws98)


```
'=====================================
'= 実行中のプロセスリストを取得
'=    (CreateToolhelp32Snapshot.bas)
'=====================================
#include "Windows.bi"

#define TH32CS_SNAPPROCESS &H2              'どの部分をスナップショットに含めたいかを示す定数宣言
#define MAX_PATH 260                        'パス名の最大長を定義

' プロセスエントリを定義する構造体
Type PROCESSENTRY32
    dwSize                  As Long
    cntUsage                As Long
    th32ProcessID           As Long
    th32DefaultHeapID       As Long
    th32ModuleID            As Long
    cntThreads              As Long
    th32ParentProcessID     As Long
    pcPriClassBase          As Long
    dwFlags                 As Long
    szExeFile               As String * MAX_PATH
End Type

' プロセスのスナップショットを取得
Declare Function Api_CreateToolhelp32Snapshot& Lib "kernel32" Alias
"CreateToolhelp32Snapshot" (ByVal dwFlag&, ByVal th32ProcessID&)

' 最初のプロセスに関する情報を取得
Declare Function Api_Process32First& Lib "kernel32" Alias "Process32First" (ByVal
hSnapshot&, lppe As PROCESSENTRY32)

' 2番目以降のプロセスに関する情報を取得する関数
Declare Function Api_Process32Next& Lib "kernel32" Alias "Process32Next" (ByVal
hSnapshot&, lppe As PROCESSENTRY32)
```

```
' オープンされているオブジェクトハンドルをクローズする
Declare Function Api_CloseHandle& Lib "kernel32" Alias "CloseHandle" (ByVal hObject&)

Var Shared List1 As Object
Var Shared Text1 As Object
Var Shared Button1 As Object

List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hSnapShot As Long
    Var pe As PROCESSENTRY32
    Var success As Long
    Var Ret As Long

    hSnapShot = Api_CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0&)

    If hSnapShot = -1 Then Exit Sub

    pe.dwSize = Len(pe)
    success = Api_Process32First(hSnapShot, pe)

    List1.ResetContent
    If success = 1 Then
        Do
            List1.AddString "# " & Right$(Str$(1000000000 + Abs(pe.th32ProcessID)), 8) & "
- " & pe.szExeFile
        Loop While Api_Process32Next(hSnapShot, pe)
    End If

    Ret = Api_CloseHandle(hSnapShot)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

実行中のプロセスを列挙(II)

実行中のプロセス一覧を列挙します。Windows9xとWindowsNT系ではアプローチが異なるため分けて実行します。
http://support.microsoft.com/default.aspx?scid=kb;ja;JP187913をF-BASIC用に書き換えています。
Process32First 最初のプロセスに関する情報を取得
Process32Next 2番目以降のプロセスに関する情報を取得
CloseHandle オープンされているオブジェクトハンドルをクローズ
OpenProcess 既存のプロセスオブジェクトのハンドルを取得
EnumProcesses プロセスの列挙
GetModuleFileNameEx ロードされている実行モジュールのフルパス名を取得
EnumProcessModules 指定されたプロセス内の各モジュールのハンドルを取得 (WindowsNT/Windows2K以降)
CreateToolhelp32Snapshot プロセスのスナップショットを取得
GetVersionEx オペレーティングシステムの種類やバージョンに関する情報を取得




```
'=======================================
'= 実行中のプロセス一覧を列挙
'= Windows9xとWindowsNT系ではアプローチが異なるため分けて実行
'=    (OpenProcess.bas)
'=======================================
#include "Windows.bi"

Type PROCESSENTRY32
    dwSize                  As Long
    cntUsage                As Long
    th32ProcessID           As Long
    th32DefaultHeapID       As Long
    th32ModuleID            As Long
    cntThreads              As Long
    th32ParentProcessID     As Long
    pcPriClassBase          As Long
    dwFlags                 As Long
    szExeFile               As String * 260
End Type

Type OSVERSIONINFO
    dwOSVersionInfoSize As Long
    dwMajorVersion    As Long
    dwMinorVersion    As Long
    dwBuildNumber     As Long
    dwPlatformId      As Long               '1 = Windows 95 • 2 = Windows NT
    szCSDVersion      As String * 128
End Type

' 最初のプロセスに関する情報を取得する関数
Declare Function Api_Process32First& Lib "Kernel32" Alias "Process32First" (ByVal
hSnapshot&, lppe As PROCESSENTRY32)

' 2番目以降のプロセスに関する情報を取得する関数
Declare Function Api_Process32Next& Lib "Kernel32" Alias "Process32Next" (ByVal
hSnapshot&, lppe As PROCESSENTRY32)

' オープンされているオブジェクトハンドルをクローズ
Declare Function Api_CloseHandle& Lib "Kernel32" Alias "CloseHandle" (ByVal hObject&)

' 既存のプロセスオブジェクトのハンドルを取得
Declare Function Api_OpenProcess& Lib "kernel32" Alias "OpenProcess" (ByVal
dwDesiredAccess&, ByVal bInheritHandle&, ByVal dwProcessID&)

' プロセスの列挙
Declare Function Api_EnumProcesses& Lib "psapi" Alias "EnumProcesses" (ByRef
lpidProcess&, ByVal cb&, ByRef cbNeeded&)

' ロードされている実行モジュールのフルパス名を取得
Declare Function Api_GetModuleFileNameEx& Lib "psapi" Alias "GetModuleFileNameExA"
(ByVal Process&, ByVal hModule&, ByVal lpFilename$, ByVal nSize&)

' 指定されたプロセス内の各モジュールのハンドルを取得(Windows NT/2000以降)
Declare Function Api_EnumProcessModules& Lib "psapi" Alias "EnumProcessModules" (ByVal
Process&, ByRef lphModule&, ByVal cb&, ByRef lpcbNeeded&)
```

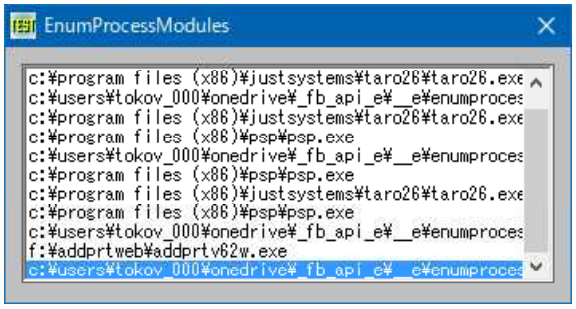
```
' プロセスのスナップショットを取得
Declare Function Api_CreateToolhelp32Snapshot& Lib "Kernel32" Alias
"CreateToolhelp32Snapshot" (ByVal dwFlag&, ByVal th32ProcessID&)

' オペレーティングシステムの種類やバージョンに関する情報を取得
Declare Function Api_GetVersionEx& Lib "Kernel32" Alias "GetVersionExA"
(lpVersionInformation As OSVERSIONINFO)

#define PROCESS_QUERY_INFORMATION &H400   '取得したハンドルをGetExitCodeProcess関数、及び
                                          GetPriorityClass関数で使用できるようにする
#define PROCESS_VM_READ &H10              '取得したハンドルをReadProcessMemory関数で使用できるようにする

#define MAX_PATH 260
#define STANDARD_RIGHTS_REQUIRED &HF0000  '標準的な権利を要求することを示す定数
#define SYNCHRONIZE &H100000              'STANDARD_RIGHTS_REQUIRED Or SYNCHRONIZE
                                          Or &HFFF
#define PROCESS_ALL_ACCESS &H1F0FFF       'すべてのアクセス権を有効にする
#define TH32CS_SNAPPROCESS &H2            'プロセス一覧のスナップショット
#define hNull 0

Var Shared List1 As Object
Var Shared Button1 As Object

List1.Attach GetDlgItem("List1") : List1.SetFontSize 12
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Function StrZToStr(s As String) As String
Function StrZToStr(s As String) As String
    StrZToStr = Left$(s, Len(s) - 1)
End Function

'=====================================
'=
'=====================================
Declare Function getVersion()
Function getVersion()
    Var osinfo As OSVERSIONINFO
    Var Ret As Integer

    osinfo.dwOSVersionInfoSize = 148
    osinfo.szCSDVersion = Space$(128)
    Ret = Api_GetVersionEx(osinfo)
    getVersion = osinfo.dwPlatformId
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    List1.ResetContent

    Select Case getVersion()
        Case 1                            'Windows 95/98
            Var f As Long
            Var sname As String
            Var hSnap As Long
            Var proc As PROCESSENTRY32

            hSnap = Api_CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0)
            If hSnap = hNull Then Exit Sub
            proc.dwSize = Len(proc)

            'プロセスを繰り返す
            f = Api_Process32First(hSnap, proc)

            Do While f
```

```
            sname = StrZToStr(proc.szExeFile)
            List1.AddString sname
            f = Api_Process32Next(hSnap, proc)
        Loop

    Case 2                                  'Windows NT
        Var cb As Long
        Var cbNeeded As Long
        Var NumElements As Long
        Var cbNeeded2 As Long
        Var ProcessIDs(300) As Long
        Var NumElements2 As Long
        Var Modules(200) As Long
        Var ModuleName As String
        Var nSize As Long
        Var hProcess As Long
        Var i As Long
        Var Ret As Long

        'プロセスIDを含む配列を取得
        cb = 8
        cbNeeded = 96
        Do While cb <= cbNeeded
            cb = cb * 2
            Ret = Api_EnumProcesses(ProcessIDs(1), cb, cbNeeded)
        Loop
        NumElements = cbNeeded / 4

        For i = 1 To NumElements

            'プロセスへのハンドルを取得
            hProcess = Api_OpenProcess(PROCESS_QUERY_INFORMATION Or PROCESS_VM_READ,
0, ProcessIDs(i))

            'プロセスハンドルを取得
            If hProcess <> 0 Then

                '指定されたモジュール配列のハンドルを取得
                Ret = Api_EnumProcessModules(hProcess, Modules(1), 200, cbNeeded2)

                'モジュールファイル名取得
                If Ret <> 0 Then
                    ModuleName = Space$(MAX_PATH)
                    nSize = 500
                    Ret = Api_GetModuleFileNameEx(hProcess, Modules(1), ModuleName,
nSize)
                    List1.AddString Left$(ModuleName, Ret)
                End If
            End If

            Ret = Api_CloseHandle(hProcess)
        Next
    End Select
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

実行中のモジュールを取得

GetForegroundWindow ユーザが操作中のウィンドウを取得

GetWindowThreadProcessId ウィンドウのプロセスIDとスレッドIDを取得
OpenProcess 既存のプロセスオブジェクトのハンドルを取得
EnumProcessModules 指定されたプロセス内の各モジュールハンドルを取得
GetModuleFileNameEx ロードされている実行モジュールのフルパス名を取得(WindowsNT4.0以降)
CloseHandle オープンされているオブジェクトハンドルをクローズ




```
'=======================================
'= 実行中のモジュールを取得
'= Windows NT/2000:Windows NT 4.0 以降
'=    (EnumProcessModules.bas)
'=======================================
#include "Windows.bi"

' ユーザーが操作中のウインドウを取得
Declare Function Api_GetForegroundWindow& Lib "user32" Alias "GetForegroundWindow" ()

' ウィンドウのプロセスIDとスレッドIDを取得
Declare Function Api_GetWindowThreadProcessId& Lib "user32" Alias
"GetWindowThreadProcessId" (ByVal hWnd&, lpdwProcessId&)

' 既存のプロセスオブジェクトのハンドルを取得
Declare Function Api_OpenProcess& Lib "kernel32" Alias "OpenProcess" (ByVal
dwDesiredAccess&, ByVal bInheritHandle&, ByVal dwProcessID&)

' 指定されたプロセス内の各モジュールのハンドルを取得(Windows NT/2000以降)
Declare Function Api_EnumProcessModules& Lib "psapi" Alias "EnumProcessModules" (ByVal
Process&, ByRef lphModule&, ByVal cb&, ByRef lpcbNeeded&)

' ロードされている実行モジュールのフルパス名を取得
Declare Function Api_GetModuleFileNameEx& Lib "psapi" Alias "GetModuleFileNameExA"
(ByVal Process&, ByVal hModule&, ByVal lpFilename$, ByVal nSize&)

' オープンされているオブジェクトハンドルをクローズ
Declare Function Api_CloseHandle& Lib "Kernel32" Alias "CloseHandle" (ByVal hObject&)

#define PROCESS_QUERY_INFORMATION &H400    '取得したハンドルをGetExitCodeProcess関数、及び
                                           GetPriorityClass関数で使用できるように
#define PROCESS_VM_READ &H10               '取得したハンドルをReadProcessMemory関数で使用でき
                                           るようにする

Var Shared List1 As Object
Var Shared Timer1 As Object

List1.Attach GetDlgItem("List1") : List1.SetFontSize 12
Timer1.Attach GetDlgItem("Timer1")

'=======================================
'=
'=======================================
Declare Function GetEXEFromHandle(hWnd As Long) As String
Function GetEXEFromHandle(hWnd As Long) As String
    Var ProcID As Long
    Var Temp As Long
    Var Modules(200) As Long
    Var File As String
    Var Process As Long
    Var Ret As Long
```

```
    If hWnd = 0 Then hWnd = Api_GetForegroundWindow()
    If Api_GetWindowThreadProcessId(hWnd, ProcID) <> 0 Then
        Process = Api_OpenProcess(PROCESS_QUERY_INFORMATION Or PROCESS_VM_READ, 0,
ProcID)
        If Process <> 0 Then
            Ret = Api_EnumProcessModules(Process, Modules(1), 200, Temp)
            If Ret <> 0 Then
                File = Space$(260)
                Ret = Api_GetModuleFileNameEx(Process, 0, File, Len(File))
                File = LCase$(Left$(File, Ret))

                GetEXEFromHandle = File
            End If
            Ret = Api_CloseHandle(Process)
        End If
    End If
End Function

'========================================
'=
'========================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Timer1.SetInterval 30
    Timer1.Enable -1
End Sub

'========================================
'=
'========================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Static Last As String
    Var File As String

    File = GetEXEFromHandle(0)
    If File <> "" And File <> Last Then
        Last = File
        List1.AddString File
        List1.SetCursel List1.GetCount - 1
    End If
End Sub

'========================================
'=
'========================================
While 1
    WaitEvent
Wend
Stop
End
```

実行ファイルの種類を取得

GetBinaryType 実行ファイルのタイプを取得表示します。

ファイル選択ボタンをクリックするとファイルオープンダイアログが開きます。
目的ファイルをクリックするとファイルの種類を表示します。

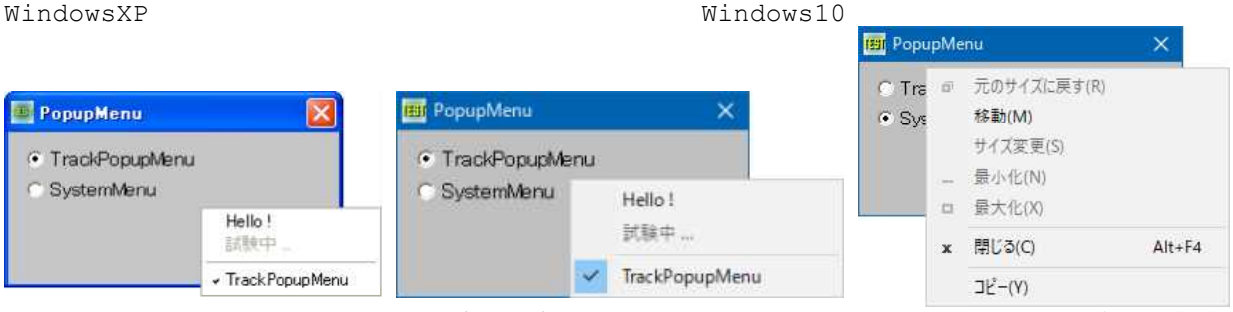
```
'=====================================
'= 実行ファイルの種類を表示
'=    (GetBinaryType.bas)
'=====================================
#include "Windows.bi"

#define SCS_32BIT_BINARY 0                  'Windowsアプリケーション
#define SCS_DOS_BINARY 1                    'DOSアプリケーション
#define SCS_OS216_BINARY 5                  '16ビットのOS2のアプリケーション
#define SCS_PIF_BINARY 3                    'PIFファイル
#define SCS_POSIX_BINARY 4                  'POSIXアプリケーション
#define SCS_WOW_BINARY 2                    '16bit Windowsアプリケーション

' 実行ファイルの種類を取得
Declare Function Api_GetBinaryType& Lib "Kernel32" Alias "GetBinaryTypeA" (ByVal
lpApplicationName$, lpBinaryType&)

Var Shared Text(3) As Object
Var Shared Button1 As Object
For i = 0 To 3
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1)))
    Text(i).SetFontSize 14
Next
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var Shared sFile As String

'=====================================
'=
'=====================================
Declare Sub BinTypeGet edecl ()
Sub BinTypeGet()
    Var BinType As Long
    Var Msg As String
    Var Ret As Long

    sFile = GetDlgItemText("Text3")
    Ret = Api_GetBinaryType(sFile, BinType)

    If Ret <> 0 Then
        Msg = "選択されたファイルは "
        Select Case BinType
            Case SCS_32BIT_BINARY
                Msg = Msg & "Win32ベースアプリケーション"
            Case SCS_DOS_BINARY
                Msg = Msg & "MS-DOSベースアプリケーション"
            Case SCS_OS216_BINARY
                Msg = Msg & "16ビットOS/2ベースアプリケーション"
            Case SCS_PIF_BINARY
                Msg = Msg & "MS-DOSベースアプリケーションを実行するPIFファイル"
            Case SCS_POSIX_BINARY
                Msg = Msg & "POSIX ベースアプリケーション"
            Case SCS_WOW_BINARY
                Msg = Msg & "16ビットWindowsベースアプリケーション"
        End Select
    Else
        Msg = "実行可能ファイルではありません！"
    End If

    Text(3).SetWindowText Msg
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    sFile = WinOpenDlg("ファイルのオープン", "C:¥*.*", "全てのﾌｧｲﾙ(*.*)", 0)
    If sFile <> Chr$(&H1B) Then
        Text(2).SetWindowText sFile
```

```
        BinTypeGet
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

指定位置にポップアップメニューの作成

カーソルで指定した位置にポップアップメニューを表示します。新たに作成するポップアップメニュー・システムメニューを選択して表示させています。
CreatePopupMenu ポップアップメニューを作成
TrackPopupMenu（TrackPopupMenuEx）指定の位置にポップアップメニューを表示
GetSystemMenu システムメニューを取得
AppendMenu メニュー末尾にメニューアイテムを追加
DestroyMenu メニューを破棄
GetCursorPos マウスカーソルの位置を取得

WindowsXP　　Windows10



Windows98ではTrackPooupMenu時左から右へ、SystemMenu時上から下へアニメーション表示されます。

TrackPopupMenuとTrackPopupMenuEx

```
Declare Function Api_TrackPopupMenu& Lib "user32" Alias "TrackPopupMenu" (ByVal hMenu&,
ByVal wFlags&, ByVal x&, ByVal y&, ByVal nReserved&, ByVal hWnd&, ByVal lprc As Any)

Declare Function Api_TrackPopupMenuEx& Lib "user32" Alias "TrackPopupMenuEx" (ByVal
hMenu&, ByVal wFlags&, ByVal x&, ByVal y&, ByVal hWnd&, ByVal lptpm As Any)

'=====================================
'= ポップアップメニューの作成
'=    (TrackPopupMenu.bas)
'=====================================
#include "Windows.bi"

#define MF_APPEND &H100                  '
#define MF_BYCOMMAND &H0                 'nPositionはメニュー項目のID
#define MF_BYPOSITION &H400              'nPositionはメニュー項目のインデックス
#define MF_CHECKED &H8                   'メニュー項目にチェックをつける
#define MF_DISABLED &H2                  'アイテムを選択不可にする
#define MF_GRAYED &H1                    'グレー表示されて選択できない
#define MF_HILITE &H80                   'メニューの項目を強調表示
#define MF_MOVE &H1000                   '
#define MF_SEPARATOR &H800               'メニュー項目はセパレータ
#define MF_STRING &H0                    '文字列
#define MF_UNHILITE &H0                  'メニューの項目を強調表示しない

#define TPM_BOTTOMALIGN &H20             'ショートカットメニューの下端を、y パラメータが指定する座標に合わせる
#define TPM_CENTERALIGN &H4              'ショートカットメニューの中心を、x パラメータが指定する座標に合わせる
#define TPM_HORNEGANIMATION &H800        '左から右へ向かってメニューのアニメーション表示
```

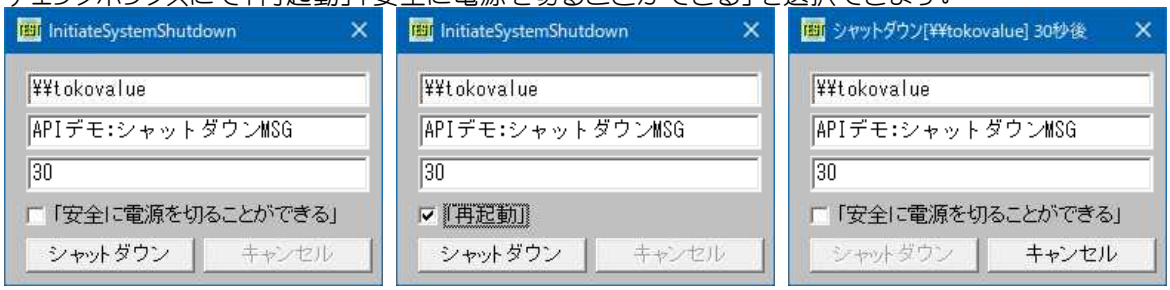
```
#define TPM_HORPOSANIMATION &H400          '右から左へ向かってメニューのアニメーション表示
#define TPM_LEFTALIGN &H0                  'ショートカットメニューの左端を、xパラメータが指定する座標に合わせる
#define TPM_LEFTBUTTON &H0                 'マウスの左ボタンでポップアップメニューからの選択が行えるようにする
#define TPM_NONOTIFY &H80                  'ユーザーがメニュー項目をクリックしたとき、通知メッセージを送らない
#define TPM_RETURNCMD &H100                '関数の戻り値として、ユーザーが選択したメニュー項目のIDを返す
#define TPM_RIGHTALIGN &H8                 'ショートカットメニューの右端を、x パラメータが指定する座標に合わせる
#define TPM_RIGHTBUTTON &H2                'マウスの右ボタンでポップアップメニューからの選択が行えるようにする
#define TPM_TOPALIGN &H0                   'ショートカットメニューの上端を、yパラメータが指定する座標に合わせる
#define TPM_VCENTERALIGN &H10              'ショートカットメニューの中心を、yパラメータが指定する座標に合わせる
#define TPM_VERNEGANIMATION &H2000         '下から上へ向かってメニューのアニメーション表示
#define TPM_VERPOSANIMATION &H1000         '上から下へ向かってメニューのアニメーション表示

Type POINTAPI
    x   As Long
    y   As Long
End Type

'ドロップダウンメニュー、サブメニュー、ショートカットメニューのいずれかを作成
Declare Function Api_CreatePopupMenu& Lib "user32" Alias "CreatePopupMenu" ()

' 指定の位置にポップアップメニューを表示
Declare Function Api_TrackPopupMenu& Lib "user32" Alias "TrackPopupMenu" (ByVal hMenu&,
ByVal wFlags&, ByVal x&, ByVal y&, ByVal nReserved&, ByVal hWnd&, ByVal lprc As Any)

' システムメニューのハンドル取得
Declare Function Api_GetSystemMenu& Lib "user32" Alias "GetSystemMenu" (ByVal hWnd&,
ByVal bRevert&)

' メニューの末尾に新しいメニュー項目を追加( InsertMenuItemを推奨)
Declare Function Api_AppendMenu& Lib "user32" Alias "AppendMenuA" (ByVal hMenu&, ByVal
wFlags&, ByVal wIDNewItem&, ByVal lpNewItem As Any)

' 指定されたメニューを破棄し、そのメニューに割り当てられていたメモリを解放
Declare Function Api_DestroyMenu& Lib "user32" Alias "DestroyMenu" (ByVal hMenu&)

' マウスカーソル(マウスポインタ)の現在の位置に相当するスクリーン座標を取得
Declare Function Api_GetCursorPos& Lib "user32" Alias "GetCursorPos" (lpPoint As
POINTAPI)

Var Shared hMenu As Long

'=====================================
'=
'=====================================
Declare Function ButtonNo bdecl () As Long
Function ButtonNo()
    ButtonNo = Val(Mid$(GetDlgRadioSelect("Radio1"), 6)) -1
End Function

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var Ret As Long

    hMenu = Api_CreatePopupMenu()

    'メニュー項目設定
    Ret = Api_AppendMenu(hMenu, MF_STRING, ByVal 0, "Hello !")
    Ret = Api_AppendMenu(hMenu, MF_GRAYED Or MF_DISABLED, ByVal 0, "試験中 ...")
    Ret = Api_AppendMenu(hMenu, MF_SEPARATOR, ByVal 0, ByVal 0)
```

```
    Ret = Api_AppendMenu(hMenu, MF_CHECKED, ByVal 0, "TrackPopupMenu")
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_MouseUp edecl (ByVal Button%, ByVal Shift%, ByVal x!, ByVal y!)
Sub MainForm_MouseUp(ByVal Button%, ByVal Shift%, ByVal x!, ByVal y!)
    Var pa As POINTAPI
    Var Ret As Long

    'マウスカーソルの位置を取得
    Ret = Api_GetCursorPos(pa)

    If ButtonNo = 0 Then

        '新規ポップアップメニュー表示
        Ret = Api_TrackPopupMenu(hMenu, TPM_LEFTALIGN Or TPM_HORPOSANIMATION, pa.x, pa.y,
0, GethWnd, ByVal 0)
    Else

        'システムメニュー表示
        Ret = Api_GetSystemMenu(GethWnd, False)
        Ret = Api_TrackPopupMenu(Ret, TPM_LEFTALIGN Or TPM_VERPOSANIMATION, pa.x, pa.y,
0, GethWnd, ByVal 0)
    End If
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_QueryClose edecl (Cancel%, ByVal Mode%)
Sub MainForm_QueryClose(Cancel%, ByVal Mode%)
    Var Ret As Long

    Ret = Api_DestroyMenu(hMenu)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

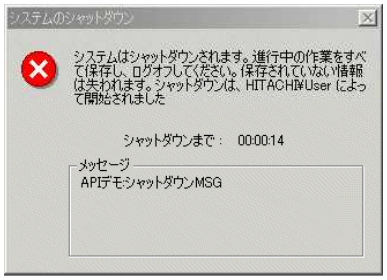
指定コンピュータのシャットダウン・中止

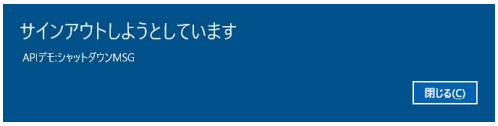
指定したコンピュータをシャットダウンおよびその処理を中止します。
InitiateSystemShutdown 指定されたコンピュータのシャットダウン処理、または再起動を設定
AbortSystemShutdown InitiateSystemShutdown関数が開始したシステムのシャットダウン処理を中止

TimeLeftで指定された時間カウントダウン表示、シャットダウン処理されます。
チェックボックスにて「再起動」「安全に電源を切ることができる」を選択できます。



¥¥Hitachi(例ではWindows2000)側では図のウィンドウが表示されます。
¥¥tokovalueでのシャットダウン及び中止の場合は右図






```
'=====================================
'= 指定コンピュータのシャットダウン･中止
'=    (InitiateSystemShutdown.bas)
'=====================================
#include "Windows.bi"

' 指定されたコンピュータのシャットダウン処理、または再起動を設定
Declare Function Api_InitiateSystemShutdown& Lib "advapi32" Alias
"InitiateSystemShutdownA" (ByVal lpMachineName$, ByVal lpMessage$, ByVal dwTimeout&,
ByVal bForceAppsClosed&, ByVal bRebootAfterShutdown&)

' InitiateSystemShutdown関数が開始したシステムのシャットダウン処理を中止
Declare Function Api_AbortSystemShutdown& Lib "advapi32" Alias "AbortSystemShutdownA"
(ByVal lpMachineName$)

Var Shared Edit(2) As Object
Var Shared Button1 As Object
Var Shared Button2 As Object
Var Shared Check1 As Object
Var Shared Timer1 As Object

For i = 0 To 2
    Edit(i).Attach GetDlgItem("Edit" & Trim$(Str$(i + 1)))
    Edit(i).SetFontSize 14
Next
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14
Check1.Attach GetDlgItem("Check1") : Check1.SetFontSize 14
Timer1.Attach GetDlgItem("Timer1")

Var Shared TimeLeft As Long                    '残り時間

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Edit(0).SetWindowText "¥¥tokovalue"
    Edit(1).SetWindowText "APIデモ:シャットダウンMSG"
    Edit(2).SetWindowText "30"

    Check1.SetWindowText "「安全に電源を切ることができる」"

    Button2.EnableWindow 0
    Timer1.SetInterval 100
    Timer1.Enable 0
End Sub

'=====================================
'= シャットダウン
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var fg As Integer
    Var Ret As Long

    fg = Check1.GetCheck
    Ret = Api_InitiateSystemShutdown(Edit(0).GetWindowText, Edit(1).GetWindowText, val
(Edit(2).GetWindowText), True, fg)
```

```
    If Ret = 0 Then
        A% = MessageBox(GetWindowText, "失敗しました！", 0, 2)
    Else
        Button1.EnableWindow 0
        Button2.EnableWindow -1
        Timer1.Enable -1
        TimeLeft = Val(Edit(2).GetWindowText)
    End If
End Sub

'=====================================
'= シャットダウンキャンセル
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Ret As Long

    Ret = Api_AbortSystemShutdown(Edit(0).GetWindowText)

    If Ret = 0 Then
        A% = MessageBox(GetWindowText, "失敗しました！", 0, 2)
    Else
        SetWindowText "キャンセルしました！"
        Button1.EnableWindow -1
        Button2.EnableWindow 0
        Timer1.Enable 0
        Wait 200
        SetWindowText "InitiateSystemShutdown"
    End If
End Sub

'=====================================
'= カウント
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    SetWindowText "シャットダウン[" & Edit(0).GetWindowText & "]" & Str$(TimeLeft) & "秒後"
    TimeLeft = TimeLeft - 1

    If TimeLeft = 0 Then
        Button1.EnableWindow -1
        Button2.EnableWindow 0
        Timer1.Enable 0
        SetWindowText "InitiateSystemShutdown"
    End If
End Sub

'=====================================
'=
'=====================================
Declare Sub Check1_on edecl ()
Sub Check1_on()
    If Check1.GetCheck = 0 Then
        Check1.SetWindowText "「安全に電源を切ることができる」"
    Else
        Check1.SetWindowText "「再起動」"
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

指定座標のRGB値取得（Ⅰ）

GradientFill 矩形、または三角形の内部をグラデーションで塗りつぶす
GetPixel 指定された座標のピクセルのRGB（赤、緑、青）値を取得

マウスでクリックした座標のRGB値を取得します。F-BasicでのGetPixel(x, y)と同じです。


```
'=======================================
'= 指定座標のRGB値取得
'=    (GetPixel.bas)
'=======================================
#include "Windows.bi"

Type GRADIENT_TRIANGLE
    Vertex1    As Long
    Vertex2    As Long
    Vertex3    As Long
End Type

Type TRIVERTEX
    X          As Long
    Y          As Long
    Red        As Integer
    Green      As Integer
    Blue       As Integer
    Alpha      As Integer
End Type

Type GRADIENT_RECT
    UpperLeft  As Long
    LowerRight As Long
End Type

' 矩形、または三角形の内部をグラデーションで塗りつぶす
Declare Function Api_GradientFill& Lib "msimg32" Alias "GradientFill" (ByVal hDC&,
pVertex As TRIVERTEX, ByVal dwNumVertex&, pMesh As GRADIENT_TRIANGLE, ByVal dwNumMesh&,
ByVal dwMode&)

' 指定された座標のピクセルのRGB値を取得
Declare Function Api_GetPixel& Lib "gdi32" Alias "GetPixel" (ByVal hDC&, ByVal X&, ByVal
Y&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

#define GRADIENT_FILL_RECT_H &H0    '
#define GRADIENT_FILL_RECT_V &H1    '
#define GRADIENT_FILL_TRIANGLE &H2  '

Var Shared hDC As Long
```

```
'====================================
'=
'====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var vert(4) As TRIVERTEX
    Var gTRi(1) As GRADIENT_TRIANGLE
    Var Ret As Long

    hDC = Api_GetDC(GethWnd)

    vert(0).X = 0
    vert(0).Y = 0
    vert(0).Red = -256
    vert(0).Green = 0
    vert(0).Blue = 0
    vert(0).Alpha = 0

    vert(1).X = 255
    vert(1).Y = 0
    vert(1).Red = 0
    vert(1).Green = -256
    vert(1).Blue = 0
    vert(1).Alpha = 0

    vert(2).X = 256
    vert(2).Y = 256
    vert(2).Red = 0
    vert(2).Green = 0
    vert(2).Blue = -256
    vert(2).Alpha = 0

    vert(3).X = 0
    vert(3).Y = 256
    vert(3).Red = -256
    vert(3).Green = -256
    vert(3).Blue = -256
    vert(3).Alpha = 0

    gTRi(0).Vertex1 = 0
    gTRi(0).Vertex2 = 1
    gTRi(0).Vertex3 = 2

    gTRi(1).Vertex1 = 0
    gTRi(1).Vertex2 = 2
    gTRi(1).Vertex3 = 3

    Ret = Api_GradientFill(hDC, vert(0), 4, gTRi(0), 2, GRADIENT_FILL_TRIANGLE)
End Sub

'====================================
'=
'====================================
Declare Sub MainForm_MouseMove edecl (ByVal Button%, ByVal Shift%, ByVal X!, ByVal Y!)
Sub MainForm_MouseMove(ByVal Button%, ByVal Shift%, ByVal X!, ByVal Y!)
    Var DotColor As String
    Var ColorMe As Long
    Var rgbRed As Integer
    Var rgbGreen As Integer
    Var rgbBlue As Integer

    ColorMe = Api_GetPixel(hDC, X, Y)
'   ColorMe = GetPixel(X, Y)

    rgbRed = Abs(ColorMe Mod &H100)
    ColorMe = Abs(ColorMe ¥ &H100)
    rgbGreen = Abs(ColorMe Mod &H100)
    ColorMe = Abs(ColorMe ¥ &H100)
    rgbBlue = Abs(ColorMe Mod &H100)
    ColorMe = RGB(rgbRed, rgbGreen, rgbBlue)
```

```
    DotColor = "GetPixel:RGB(" & Format$(rgbRed, "###") & ", " & Format$(rgbGreen, "###")
& ", " & Format$(rgbBlue, "###") & ")"
    SetWindowText DotColor
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

指定座標のRGB値を取得(II)

指定座標のRGB値を取得します。
GetCursorPos マウスカーソル(マウスポインタ)の現在の位置に相当するスクリーン座標を取得
GetPixel 指定された座標のピクセルのRGB値を取得
GetWindowDC ウィンドウ全体のデバイスコンテキストを取得
ReleaseDC デバイスコンテキストを解放

例ではウインドウ全体のデバイスコンテキストを取得し、カーソル位置のRGB値を取得し、その値および色を表示しています。

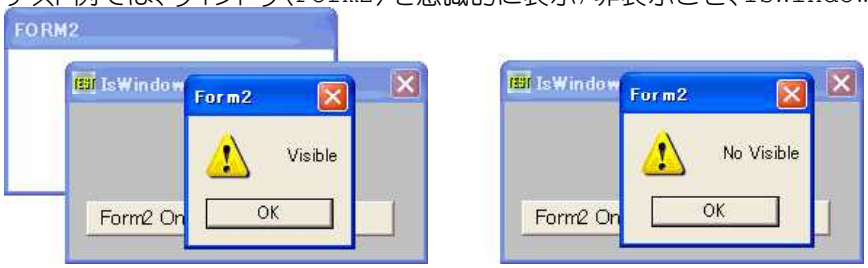

```
'=====================================
'= 指定座標のRGB値を取得
'=    (GetPixel2.bas)
'=====================================
#include "Windows.bi"

Type POINTAPI
    x   As Long
    y   As Long
End Type

' マウスカーソル(マウスポインタ)の現在の位置に相当するスクリーン座標を取得
Declare Function Api_GetCursorPos& Lib "user32" Alias "GetCursorPos" (lpPoint As
POINTAPI)

' 指定された座標のピクセルのRGB値を取得
Declare Function Api_GetPixel& Lib "gdi32" Alias "GetPixel" (ByVal hDC&, ByVal X&, ByVal
Y&)

' ウィンドウ全体のデバイスコンテキストを取得。0(NULL)を指定すると、スクリーン全体のデバイスコンテキストを取
得
Declare Function Api_GetWindowDC& Lib "user32" Alias "GetWindowDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

Var Shared Timer1 As Object
Var Shared Picture1 As Object
Var Shared Text(6) As Object

Timer1.Attach GetDlgItem("Timer1")
Picture1.Attach GetDlgItem("Picture1")
For i = 0 To 6
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1)))
```

```
    Text(i).SetFontSize 14
Next

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Timer1.SetInterval 10
    Timer1.Enable -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var pAPI As POINTAPI
    Var Col As String
    Var lColor As Long
    Var lDC As Long
    Var Ret As Long

    lDC = Api_GetWindowDC(0)

    Ret = Api_GetCursorPos(pAPI)
    lColor = Api_GetPixel(lDC, pAPI.x, pAPI.y)

    Picture1.SetBackColor lColor
    Picture1.cls

    Col = Right$("000000" & Hex$(lColor), 6)
    Text(1).SetWindowText Right$(Col, 2)
    Text(2).SetWindowText Mid$(Col, 3, 2)
    Text(3).SetWindowText Left$(Col, 2)

    Text(4).SetWindowText Trim$(Str$(Val("&H" & Right$(Col, 2))))
    Text(5).SetWindowText Trim$(Str$(Val("&H" & Mid$(Col, 3, 2))))
    Text(6).SetWindowText Trim$(Str$(Val("&H" & Left$(Col, 2))))
    Ret = Api_ReleaseDC(0, lDC)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

指定されたウィンドウの表示状態を調べる

指定したウィンドウが表示されているかどうかを調べます。
IsWindowVisible 指定されたウィンドウの表示状態を調べる

テスト例では、ウィンドウ(Form2)を意識的に表示/非表示させ、IsWindowVisibleの結果を確認しています。

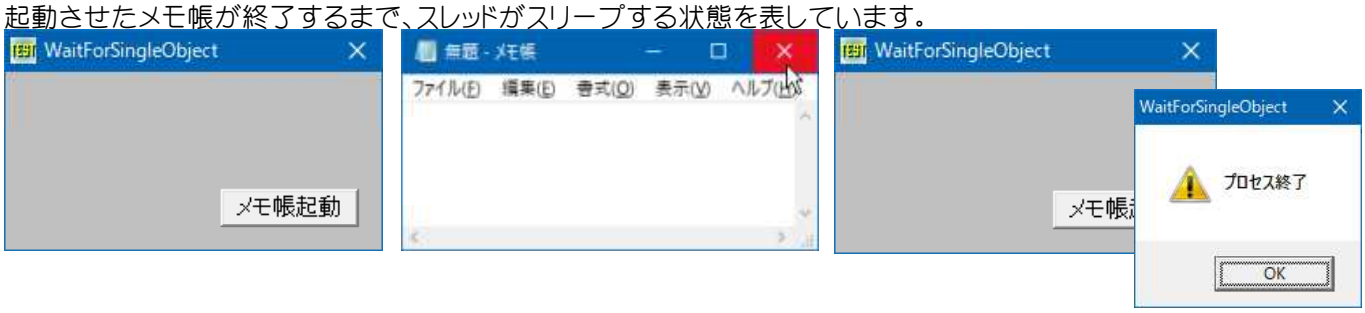


```
'====================================
'= 指定されたウィンドウの表示状態を調べる
'=    (IsWindowVisible.bas)
'====================================
#include "Windows.bi"

' 指定されたウィンドウの表示状態を調べる
Declare Function Api_IsWindowVisible& Lib "user32" Alias "IsWindowVisible" (ByVal hWnd&)

Var Shared Button1 As Object
Var Shared Button2 As Object

Button1.Attach getDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach getDlgItem("Button2") : Button2.SetFontSize 14

Var Shared Form2 As Object
Var Shared FLG As Integer

'====================================
'=
'====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    If Not (CheckObject(Form2)) Then
        Form2.CreateWindow "Form2", -1
    End If
End Sub

'====================================
'= Form2の表示/非表示
'====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    FLG = 1 - FLG

    If FLG = 0 Then
        Form2.ShowWindow -1
    Else
        Form2.ShowWindow 0
    End If
End Sub

'====================================
'= Form2の表示状態を知る
'====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Ret As Long

    Ret = Api_IsWindowVisible(Form2.GethWnd)

    If Ret <> 0 Then
        A% = MessageBox("Form2", "Visible", 0, 2)
    Else
        A% = MessageBox("Form2", "No Visible", 0, 2)
    End If
End Sub

'====================================
'=
'====================================
While 1
    WaitEvent
Wend
Stop
End
```

指定された時間スレッドをスリープ

CreateProcess プロセスの起動
GetExitCodeProcess 指定プロセスの終了コードを取得
WaitForSingleObject 指定された時間が経過するまでスレッドをスリープ
CloseHandle オープンされているオブジェクトハンドルをクローズ

起動させたメモ帳が終了するまで、スレッドがスリープする状態を表しています。




```
'=======================================
'= 指定された時間スレッドをスリープ
'=    (WaitForSingleObject3.bas)
'=======================================
#include "Windows.bi"

Type STARTUPINFO
    cb                As Long
    lpReserved        As Long
    lpDesktop         As Long
    lpTitle           As Long
    dwX               As Long
    dwY               As Long
    dwXSize           As Long
    dwYSize           As Long
    dwXCountChars     As Long
    dwYCountChars     As Long
    dwFillAttribute   As Long
    dwFlags           As Long
    wShowWindow       As Integer
    cbReserved2       As Integer
    lpReserved2       As Long
    hStdInput         As Long
    hStdOutput        As Long
    hStdError         As Long
End Type

Type PROCESS_INFORMATION
    hProcess          As Long
    hThread           As Long
    dwProcessID       As Long
    dwThreadID        As Long
End Type

' プロセスの起動
Declare Function Api_CreateProcess& Lib "kernel32" Alias "CreateProcessA" (ByVal
lpAppName&, ByVal lpCmdLin$, ByVal lpPrcAttr&, ByVal lpAttr&, ByVal bHand&, ByVal
dwFlag&, ByVal lpEnv&, ByVal lpDir&, lpInf As STARTUPINFO, lpPrcInf As
PROCESS_INFORMATION)

' 指定プロセスの終了コードを取得
Declare Function Api_GetExitCodeProcess& Lib "Kernel32" Alias "GetExitCodeProcess"
(ByVal hProcess&, lpExitCode&)

' 指定されたカーネルオブジェクトがシグナル状態になるか、指定された時間が経過するまでスレッドをスリープ
Declare Function Api_WaitForSingleObject& Lib "Kernel32" Alias "WaitForSingleObject"
(ByVal hHandle&, ByVal dwMilliseconds&)

' オープンされているオブジェクトハンドルをクローズ
Declare Function Api_CloseHandle& Lib "Kernel32" Alias "CloseHandle" (ByVal hObject&)
```

```
#define NORMAL_PRIORITY_CLASS &H20          '通常クラス(一般的なプロセス)
#define INFINITE -1                         '無限に待つ

Var Shared Button1 As Object

Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Function ExecCmd(cmdline As String)
Function ExecCmd(cmdline As String)
    Var si As STARTUPINFO
    Var pi As PROCESS_INFORMATION
    Var Ret As Long

    'STARTUPINFO構造体初期化
    si.cb = Len(si)

    '指定プロセス(メモ帳)を起動
    Ret = Api_CreateProcess(ByVal 0, cmdline, 0, 0, 1, NORMAL_PRIORITY_CLASS, 0, ByVal 0,
si, pi)

    '指定プロセスの終了を待つ
    Ret = Api_WaitForSingleObject(pi.hProcess, INFINITE)

    '指定プロセスの終了コードを取得
    Ret = Api_GetExitCodeProcess(pi.hProcess, Ret)

    'オブジェクトハンドルをクローズ
    Ret = Api_CloseHandle(pi.hThread)
    Ret = Api_CloseHandle(pi.hProcess)
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    Ret = ExecCmd("notepad.exe")
    A% = MessageBox(GetWindowtext, "プロセス終了", 0, 2)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

指定された水平および垂直のオフセット分だけ移動

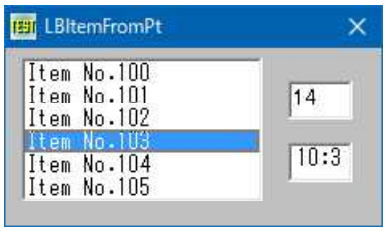
OffsetWindowOrgEx 指定されたデバイスコンテキストのウィンドウ原点を、指定された水平および垂直のオフセット分だけ移動
GetWindowOrgEx 指定されたデバイスコンテキストのウィンドウ原点のx座標とy座標を取得
GetDC 指定されたウィンドウのデバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放


```
'=====================================
'= 指定された水平および垂直のオフセット分だけ移動
'=    (OffsetWindowOrgEx.bas)
'=====================================
#include "Windows.bi"

Type POINTAPI
    x   As Long
    y   As Long
End Type

' 指定されたデバイスコンテキストのウィンドウ原点を、指定された水平および垂直のオフセット分だけ移動
Declare Function Api_OffsetWindowOrgEx& Lib "gdi32" Alias "OffsetWindowOrgEx" (ByVal
hDC&, ByVal nX&, ByVal nY&, lpPoint As POINTAPI)

' 指定されたデバイスコンテキストのウィンドウ原点のx座標とy座標を取得
Declare Function Api_GetWindowOrgEx& Lib "gdi32" Alias "GetWindowOrgEx" (ByVal hDC&,
lpPoint As POINTAPI)

' 指定されたウィンドウのデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

Var Shared Picture1 As Object
Var Shared Button1 As Object
Var Shared Bitmap As Object

BitmapObject Bitmap

Picture1.Attach GetDlgItem("Picture1")
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var Shared hDC As Long

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    hDC = Api_GetDC(Picture1.GethWnd)

    Bitmap.LoadFile "Bike1.bmp"
    Picture1.DrawBitmap Bitmap, 0, 3
    Bitmap.DeleteObject
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var pa As POINTAPI
    Var Ret As Long

    For i% = 1 To 10
        Ret = Api_OffsetWindowOrgEx(hDC, i%, 3, pa)
        Ret = Api_GetWindowOrgEx(hDC, pa)

        Picture1.Cls
        Bitmap.LoadFile "Bike1.bmp"
```

```
        Picture1.DrawBitmap Bitmap, pa.x, 3
        Bitmap.DeleteObject
        Wait 2
    Next
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
    Var Ret As Long

    Ret = Api_ReleaseDC(Picture1.GethWnd, hDC)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

指定された点の項目のリストインデックスを取得

GetCursorPos マウスカーソル(マウスポインタ)の現在の位置に相当するスクリーン座標を取得
ClientToScreen 指定されたウィンドウ上の点の座標を、クライアント領域の座標からスクリーン座標に変換
ScreenToClient マウスカーソルの現在の位置に相当するスクリーン座標を取得し、クライアント座標に変換
LBItemFromPt 指定された点にある項目のインデックスを取得



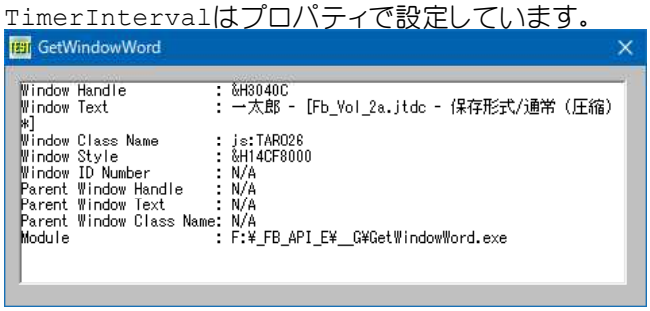

```
'=====================================
'= 指定された点の項目のリストインデックスを取得
'=     (LBItemFromPt.bas)
'=====================================
#include "Windows.bi"

Type POINTAPI
    X   As Long
    Y   As Long
End Type

' マウスカーソル(マウスポインタ)の現在の位置に相当するスクリーン座標を取得
Declare Function Api_GetCursorPos& Lib "user32" Alias "GetCursorPos" (lpPoint As
POINTAPI)

' 指定されたウィンドウ上の点の座標を、クライアント領域の座標からスクリーン座標に変換
Declare Function Api_ClientToScreen& Lib "user32" Alias "ClientToScreen" (ByVal hWnd&,
lpPoint As POINTAPI)

' マウスカーソルの現在の位置に相当するスクリーン座標を取得し、クライアント座標に変換
Declare Function Api_ScreenToClient& Lib "user32" Alias "ScreenToClient" (ByVal hWnd&,
lpPoint As POINTAPI)

' 指定された点にある項目のインデックスを取得
Declare Function Api_LBItemFromPt& Lib "comctl32" Alias "LBItemFromPt" (ByVal hWnd&,
ByVal ptx&, ByVal pty&, ByVal bAutoScroll&)
```

```
#define vbRightButton 2                    '右ボタンクリック

Var Shared Edit1 As Object
Var Shared Text1 As Object
Var Shared List1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14

Var Shared fs As Integer

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var i As Integer

    fs = 14

    List1.Attach GetDlgItem("List1")
    List1.SetWindowSize 150, 96
    List1.SetFontSize fs

    For i = 100 To 120
        List1.AddString "Item No." & Trim$(Str$(i))
    Next
End Sub

'=====================================
'=
'=====================================
Declare Sub List1_Click edecl ()
Sub List1_Click()
    Var hDC As Long
    Var pa As POINTAPI
    Var Ret As Long

    Ret = Api_GetCursorPos(pa)
    Ret = Api_ScreenToClient(List1.GethWnd, pa)
    Ret = Api_LBItemFromPt(List1.GethWnd, pa.X, pa.Y, 1)
    Text1.SetWindowText Trim$(Str$(pa.X ¥ (fs / 2))) & ":" & Trim$(Str$(pa.Y ¥ fs))
End Sub

'=====================================
'=
'=====================================
Declare Sub Edit1_Change edecl ()
Sub Edit1_Change()
    List1.Detach
    Text1.SetWindowText ""

    List1.Attach GetDlgItem("List1")
    List1.SetWindowSize 150, 96
    List1.SetFontSize fs
    fs = Val(Edit1.GetWindowText)
    List1.SetFontSize fs
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

指定したウィンドウの情報を取得

マウスで指定したウィンドウの情報を取得します。
GetCursorPos マウスカーソルのスクリーン座標を取得
WindowFromPoint 指定の座標位置にあるウィンドウハンドルを取得
GetModuleFileName ロードされている実行モジュールのフルパス名を取得
GetWindowWord ウィンドウに関連付けている補足データ域からワード値を取得
GetWindowLong 指定されたウィンドウに関しての情報を取得
GetParent 指定された子ウィンドウの親ウィンドウまたはオーナーウィンドウハンドルを返す
GetClassName ウィンドウクラス名を取得
GetWindowText ウィンドウタイトル文字列を取得

TimerIntervalはプロパティで設定しています。




```
'=====================================
'= 指定したウィンドウの情報を取得
'=    (GetWindowWord.bas)
'=====================================
#include "Windows.bi"

Type POINTAPI
    x   As Long
    y   As Long
End Type

' マウスカーソル（マウスポインタ）の現在の位置に相当するスクリーン座標を取得
Declare Function Api_GetCursorPos& Lib "user32" Alias "GetCursorPos" (lpPoint As
POINTAPI)

' 指定の座標位置にあるウィンドウハンドルを取得
Declare Function Api_WindowFromPoint& Lib "user32" Alias "WindowFromPoint" (ByVal
xPoint&, ByVal yPoint&)

' ロードされている実行モジュールのフルパス名を取得
Declare Function Api_GetModuleFileName& Lib "Kernel32" Alias "GetModuleFileNameA" (ByVal
hModule&, ByVal lpFileName$, ByVal nSize&)

' ウィンドウに関連付けている補足データ域からワード値を取得
Declare Function Api_GetWindowWord% Lib "user32" Alias "GetWindowWord" (ByVal hWnd&,
ByVal nIndex&)

' 指定されたウィンドウに関しての情報を取得。また、拡張ウィンドウメモリから、指定されたオフセットにある32ビット値
を取得することもできる
Declare Function Api_GetWindowLong& Lib "user32" Alias "GetWindowLongA" (ByVal hWnd&,
ByVal nIndex&)

' 指定された子ウィンドウの親ウィンドウまたはオーナーウィンドウのハンドルを返す
Declare Function Api_GetParent& Lib "user32" Alias "GetParent" (ByVal hWnd&)

' ウィンドウのクラス名を取得
Declare Function Api_GetClassName& Lib "user32" Alias "GetClassNameA" (ByVal hWnd&,
ByVal lpClassName$, ByVal nMaxCount&)

' ウィンドウのタイトル文字列を取得
Declare Function Api_GetWindowText& Lib "user32" Alias "GetWindowTextA" (ByVal hWnd&,
ByVal lpString$, ByVal cch&)

#define GWW_HINSTANCE -6
#define GWW_ID -12
```

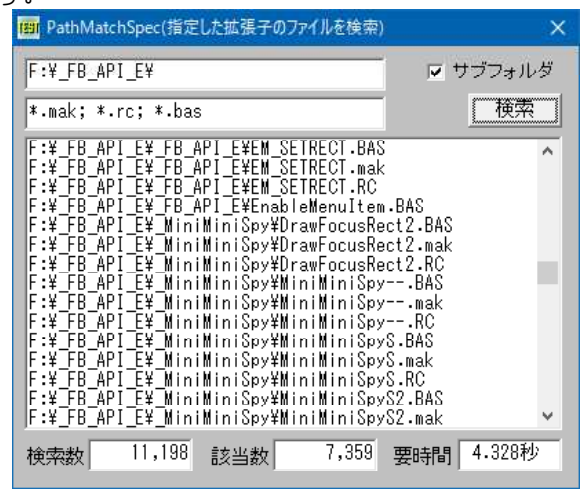
```
#define GWL_STYLE -16                       'アプリケーションのインスタンスハンドル
#define vbCrLf (Chr$(13) & Chr$(10))        'キャリッジリターンとラインフィード(¥r¥n)

Var Shared Timer1 As Object
Var Shared Text1 As Object

Timer1.Attach GetDlgItem("Timer1")
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 12

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var pt As POINTAPI
    Var ptx As Long
    Var pty As Long
    Var sWindowText As String * 100
    Var sClassName As String * 100
    Var hWndOver As Long
    Var hWndParent As Long
    Var sParentClassName As String * 100
    Var wID As Long
    Var lWindowStyle As Long
    Var hInstance As Long
    Var sParentWindowText As String * 100
    Var sModuleFileName As String * 100
    Static hWndLast As Long
    Var txt As String
    Var Ret As Long

    Ret = Api_GetCursorPos(pt)
    ptx = pt.x
    pty = pt.y

    hWndOver = Api_WindowFromPoint(ptx, pty)

    If hWndOver <> hWndLast Then
        hWndLast = hWndOver
        Cls
        txt = txt & "Window Handle          : &&H" & Hex$(hWndOver) & vbCrLf

        Ret = Api_GetWindowText(hWndOver, sWindowText, 100)
        txt = txt & "Window Text            : " & Left$(sWindowText, Ret) & vbCrLf

        Ret = Api_GetClassName(hWndOver, sClassName, 100)
        txt = txt & "Window Class Name      : " & Left$(sClassName, Ret) & vbCrLf

        lWindowStyle = Api_GetWindowLong(hWndOver, GWL_STYLE) ' Window Style
        txt = txt & "Window Style           : &&H" & Hex$(lWindowStyle) & vbCrLf

        'パレントウィンドウのハンドルを取得
        hWndParent = Api_GetParent(hWndOver)

        'ウィンドウの情報取得
        If hWndParent <> 0 Then

            'WindowID取得
            wID = Api_GetWindowWord(hWndOver, GWW_ID)
            txt = txt & "Window ID Number       : &&H" & Hex$(wID) & vbCrLf
            txt = txt & "Parent Window Handle   : &&H" & Hex$(hWndParent) & vbCrLf

            'ParentWindowのテキスト取得
            Ret = Api_GetWindowText(hWndParent, sParentWindowText, 100)
            txt = txt & "Parent Window Text     : " & Left$(sParentWindowText, Ret) & vbCrLf

            'クラス名取得
            Ret = Api_GetClassName(hWndParent, sParentClassName, 100)
            txt = txt & "Parent Window Class Name: " & Left$(sParentClassName, Ret) & vbCrLf
        Else
```

```
            'Parentの無いとき
            txt = txt & "Window ID Number        : N/A" & vbCrLf
            txt = txt & "Parent Window Handle    : N/A" & vbCrLf
            txt = txt & "Parent Window Text      : N/A" & vbCrLf
            txt = txt & "Parent Window Class Name: N/A" & vbCrLf
        End If

        'ウィンドウインスタンス取得
        hInstance = Api_GetWindowWord(hWndOver, GWW_HINSTANCE)

        'モジュールファイル名取得
        Ret = Api_GetModuleFileName(hInstance, sModuleFileName, 100)
        txt = txt & "Module                  : " &  Left$(sModuleFileName, Ret) & vbCrLf

        '結果表示
        Text1.SetWindowtext txt
    End If
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

指定した拡張子のファイルを検索

指定した拡張子のファイルを、フォルダおよびサブフォルダから検索表示します。
FindClose ファイル検索ハンドルをクローズ
FindFirstFile 指定したファイル名に一致するファイルやディレクトリを検索
FindNextFile FindFirstFileで検出したファイルの次を検出
GetTickCount システムが起動してからの経過時間を取得
PathMatchSpec パスが検索文字列に一致するかどうかの判断
lstrlen 指定された文字列のバイトまたは文字の長さを返す

複数の拡張子を検索する場合は、「;」で接続します。




```
'=======================================
'= 指定した拡張子のファイルを検索
'=    (PathMatchSpec.bas)
'=======================================
#include "Windows.bi"

#define MAX_PATH 260
#define INVALID_HANDLE_VALUE -1             '見つからない場合
#define vbDot 46                            '「.」
#define vbDirectory 16                      'フォルダ
```

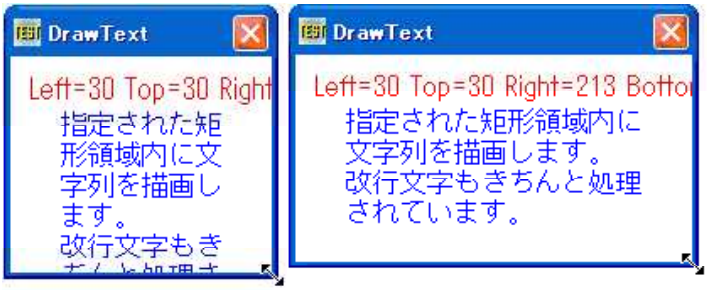
```
#define vbBackslash "¥"
#define ALL_FILES "*.*"

Type FILETIME
    dwLowDateTime     As Long
    dwHighDateTime    As Long
End Type

Type WIN32_FIND_DATA
    dwFileAttributes As Long
    ftCreationTime   As FILETIME
    ftLastAccessTime As FILETIME
    ftLastWriteTime  As FILETIME
    nFileSizeHigh    As Long
    nFileSizeLow     As Long
    dwReserved0      As Long
    dwReserved1      As Long
    cFileName        As String * MAX_PATH
    cAlternate       As String * 14
End Type

' ファイル検索ハンドルをクローズ
Declare Function Api_FindClose& Lib "Kernel32" Alias "FindClose" (ByVal hFindFile&)

' 指定したファイル名に一致するファイルやディレクトリを検索
Declare Function Api_FindFirstFile& Lib "Kernel32" Alias "FindFirstFileA" (ByVal
lpFileName$, lpFindFileData As WIN32_FIND_DATA)

' FindFirstFile()関数で検出したファイルの次を検出
Declare Function Api_FindNextFile& Lib "Kernel32" Alias "FindNextFileA" (ByVal
hFindFile&, lpFindFileData As WIN32_FIND_DATA)

' システムが起動してからの経過時間を取得
Declare Function Api_GetTickCount& Lib "Kernel32" Alias "GetTickCount" ()

' パスが検索文字列に一致するかどうか判定
Declare Function Api_PathMatchSpec& Lib "shlwapi" Alias "PathMatchSpecA" (ByVal
pszFile$, ByVal pszSpec$)

' 指定された文字列のバイトまたは文字の長さを返す
Declare Function Api_lstrlen& Lib "Kernel32" Alias "lstrlenA" (ByVal lpString$)

Var Shared bRecurse As Integer
Var Shared nCount As Long
Var Shared nSearched As Long
Var Shared sFileNameExt As String
Var Shared sFileRoot As String

Var Shared Edit(4) As Object
Var Shared Text(2) As Object
Var Shared List1 As Object
Var Shared Check1 As Object
Var Shared Button1 As Object

For i = 0 To 4
    If i < 3 Then
        Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1)))
        Text(i).SetFontSize 14
    End If
    Edit(i).Attach GetDlgItem("Edit" & Trim$(Str$(i + 1)))
    Edit(i).SetFontSize 14
Next
List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
Check1.Attach GetDlgItem("Check1") : Check1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Function TrimNull(sFileName As String) As String
```

```
Function TrimNull(sFileName As String) As String
    TrimNull = Left$(sFileName, Api_lstrlen(sFileName))
End Function

'=====================================
'= 検索部
'=====================================
Declare Sub SearchForFiles(sRoot As String)
Sub SearchForFiles(sRoot As String)
    Var wfd As WIN32_FIND_DATA
    Var hFile As Long

    hFile = Api_FindFirstFile(sRoot & ALL_FILES, wfd)

    If hFile <> INVALID_HANDLE_VALUE Then
        Do
            If (wfd.dwFileAttributes And vbDirectory) Then
                If Asc(wfd.cFileName) <> vbDot Then
                    If bRecurse = 1 Then
                        SearchForFiles sRoot & TrimNull(wfd.cFileName) & vbBackslash
                    End If
                End If
            Else
                If Api_PathMatchSpec(wfd.cFileName, sFileNameExt) <> 0 Then
                    nCount = nCount + 1
                    List1.AddString sRoot & TrimNull(wfd.cFileName)
                End If
            End If
            nSearched = nSearched + 1
        Loop While Api_FindNextFile(hFile, wfd)
    End If
    Ret = Api_FindClose(hFile)
End Sub

'=====================================
'=「¥」が無い場合付加
'=====================================
Declare Function QualifyPath(sPath As String) As String
Function QualifyPath(sPath As String) As String
    If Right$(sPath, 1) <> vbBackslash Then
        QualifyPath = sPath & vbBackslash
    Else
        QualifyPath = sPath
    End If
End Function

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
   Edit(0).SetWindowText "D:¥_FB_API_E¥"
   Edit(1).SetWindowtext "*.mak; *.rc; *.bas"
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var tmStart As Single
    Var tmEnd As Single
    Var Ret As Long

    For i% = 2 To 4 : Edit(i%).SetWindowText "" : Next
    List1.Resetcontent
    sFileRoot = QualifyPath(Edit(0).GetWindowText)
    sFileNameExt = Edit(1).GetWindowText
    bRecurse = Check1.GetCheck
    nCount = 0
```

```
    nSearched = 0

    tmStart = Api_GetTickCount()
    SearchForFiles sFileRoot
    tmEnd = Api_GetTickCount()

    Edit(2).SetWindowText Format$(nSearched, "##,###,###")
    Edit(3).SetWindowText Format$(nCount, "##,###,###")
    Edit(4).SetWindowText Str$((tmEnd - tmStart) / 1000) & "秒"
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

指定した矩形領域に文字列を描画（Ⅰ）

SetTextColor デバイスコンテキストの文字色を変更
GetClientRect ウィンドウのクライアント領域の座標を取得
TextOut 文字を描画
DrawText 文字列を指定領域に出力
SetBkMode バックグラウンドの塗りつぶしモード設定
GetDC 指定されたウィンドウのデバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放
DT_WORDBREAK(&H10) テキストを複数行で表示。折り返しは自動的に行われる
TRANSPARENT(1) 背景色を設定しない




```
'=====================================
'= 指定した矩形領域に文字列を描画（Ⅰ）
'=    (DrawText2.bas)
'=====================================
#include "Windows.bi"

Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

#define DT_WORDBREAK &H10                   'テキストを複数行で表示。折り返しは自動的に行われる
#define TRANSPARENT 1                       '背景色を設定しない

' デバイスコンテキストの文字色を変更
Declare Function Api_SetTextColor& Lib "gdi32" Alias "SetTextColor" (ByVal hDC&, ByVal
crColor&)

' ウィンドウのクライアント領域の座標を取得
Declare Function Api_GetClientRect& Lib "user32" Alias "GetClientRect" (ByVal hWnd&,
lpRect As RECT)
```

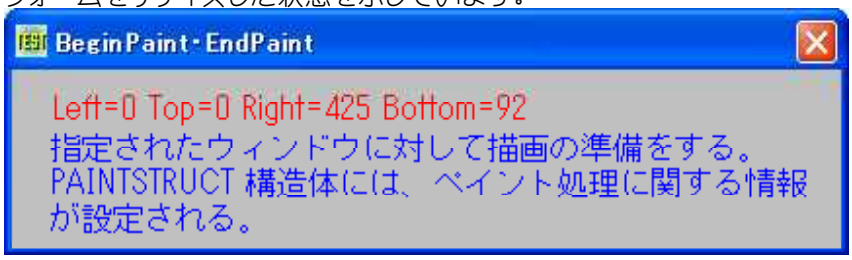
```
' 文字を描画
Declare Function Api_TextOut& Lib "gdi32" Alias "TextOutA" (ByVal hDC&, ByVal nXStart&,
ByVal nYStart&, ByVal lpString$, ByVal cbString&)

' 文字列を指定領域に出力
Declare Function Api_DrawText& Lib "user32" Alias "DrawTextA" (ByVal hDC&, ByVal lpStr$,
ByVal nCount&, lpRect As RECT, ByVal wFormat&)

' バックグラウンドの塗りつぶしモード設定
Declare Function Api_SetBkMode& Lib "gdi32" Alias "SetBkMode" (ByVal hDC&, ByVal
iBkMode&)

' 指定されたウィンドウのデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

'=======================================
'=
'=======================================
Declare Sub MainForm_Resize edecl ()
Sub MainForm_Resize()
    Var hDC As Long
    Var rct As RECT
    Var txt As String
    Var Ret As Long

    hDC = Api_GetDC(GethWnd)
    Ret = Api_GetClientRect(GethWnd, rct)

    rct.Left = 30
    rct.Top = 30
    rct.Right = GetWidth - 30
    rct.Bottom = GetHeight - 30

    txt = "Left=" & Trim$(Str$(rct.Left)) & " Top=" & Trim$(Str$(rct.Top)) & " Right=" &
Trim$(Str$(rct.Right)) & " Bottom=" & Trim$(Str$(rct.Bottom))

    Ret = Api_SetTextColor(hDC, RGB(255, 0, 0))
    Ret = Api_SetBkMode(hDC, TRANSPARENT)
    Ret = Api_TextOut(hDC, 10, 10, txt, Len(txt))

    txt = "指定された矩形領域内に文字列を描画します。" & Chr$(13, 10) & "改行文字もきちんと処理されて
います。"

    Ret = Api_SetTextColor(hDC, RGB(0, 0, 255))
    Ret = Api_SetBkMode(hDC, TRANSPARENT)
    Ret = Api_DrawText(hDC, txt, Len(txt), rct, DT_WORDBREAK)

    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

指定した矩形領域に文字列を描画（II）

SetTextColor デバイスコンテキストの文字色を変更
GetClientRect ウィンドウのクライアント領域の座標を取得
TextOut 文字を描画

DrawText 文字列を指定領域に出力
SetBkMode バックグラウンドの塗りつぶしモード設定
BeginPaint 指定されたウィンドウに対して描画の準備
EndPaint 指定されたウィンドウ内の描画の終わりを示す
GetDC 指定されたウィンドウのデバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放
DT_WORDBREAK(&H10) テキストを複数行で表示。折り返しは自動的に行われる
TRANSPARENT(1) 背景色を設定しない

フォームをリサイズした状態を示しています。




```
'=====================================
'= 指定した矩形領域に文字列を描画(II)
'=    (BeginPaint.bas)
'=====================================
#include "Windows.bi"

Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

Type PAINTSTRUCT
    fhDC        As Long                     'デバイスコンテキストを識別
    fErase      As Long                     'このメンバが 1 のときは、バックグラウンドを再描画
    rcPaint     As RECT                     '更新(再描画)する矩形座標を指定するRECT構造体
    fRestore    As Long                     '予約されているメンバ
    fIncUpdate  As Long                     '予約されているメンバ
    rgbReserved(31) As Byte                 '予約されているメンバ
End Type

#define DT_WORDBREAK &H10                   'テキストを複数行で表示。折り返しは自動的に行われる
#define TRANSPARENT 1                       '背景色を設定しない

' デバイスコンテキストの文字色を変更
Declare Function Api_SetTextColor& Lib "gdi32" Alias "SetTextColor" (ByVal hDC&, ByVal
crColor&)

' ウィンドウのクライアント領域の座標を取得
Declare Function Api_GetClientRect& Lib "user32" Alias "GetClientRect" (ByVal hWnd&,
lpRect As RECT)

' 文字を描画
Declare Function Api_TextOut& Lib "gdi32" Alias "TextOutA" (ByVal hDC&, ByVal nXStart&,
ByVal nYStart&, ByVal lpString$, ByVal cbString&)

' 文字列を指定領域に出力
Declare Function Api_DrawText& Lib "user32" Alias "DrawTextA" (ByVal hDC&, ByVal lpStr$,
ByVal nCount&, lpRect As RECT, ByVal wFormat&)
```

```
' バックグラウンドの塗りつぶしモード設定
Declare Function Api_SetBkMode& Lib "gdi32" Alias "SetBkMode" (ByVal hDC&, ByVal
iBkMode&)

' 指定されたウィンドウに対して描画の準備
Declare Function Api_BeginPaint& Lib "user32" Alias "BeginPaint" (ByVal hWnd&, lpPaint As
PAINTSTRUCT)

' 指定されたウィンドウ内の描画の終わりを示す
Declare Function Api_EndPaint& Lib "user32" Alias "EndPaint" (ByVal hWnd&, lpPaint As
PAINTSTRUCT)

' 指定されたウィンドウのデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

'=====================================
'=
'=====================================
Declare Sub MainForm_Resize edecl ()
Sub MainForm_Resize()
    Var hDC As Long
    Var hWnd As Long
    Var rct As RECT
    Var ps As PAINTSTRUCT
    Var txt As String
    Var Ret As Long

    Ret = Api_GetClientRect(GethWnd, rct)
    hDC = Api_GetDC(GethWnd)

    ps.fhDC = hDC
    ps.fErase = 1

    txt = "Left=" & Trim$(Str$(rct.Left)) & " Top=" & Trim$(Str$(rct.Top)) & " Right=" &
Trim$(Str$(rct.Right)) & " Bottom=" & Trim$(Str$(rct.Bottom))

    Ret = Api_BeginPaint(GethWnd, ps)

    Ret = Api_SetTextColor(hDC, RGB(255, 0, 0))
    Ret = Api_SetBkMode(hDC, TRANSPARENT)
    Ret = Api_TextOut(hDC, 20, 10, txt, Len(txt))

    rct.Left = 20
    rct.Top = 30
    rct.Right = GetWidth - 20
    rct.Bottom = GetHeight - 20

    txt = "指定されたウィンドウに対して描画の準備をする。" & Chr$(13, 10) & "PAINTSTRUCT 構造体には、
ペイント処理に関する情報が設定される。"

    Ret = Api_SetTextColor(hDC, RGB(0, 0, 255))
    Ret = Api_SetBkMode(hDC, TRANSPARENT)
    Ret = Api_DrawText(hDC, txt, Len(txt), rct, DT_WORDBREAK)

    Ret = Api_EndPaint(GethWnd, ps)
    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

指定したスレッドの優先順位を設定

GetPriorityClass 指定のプロセスのプライオリティクラスを取得
SetPriorityClass 指定のプロセスのプライオリティクラスを設定
GetThreadPriority 指定したスレッドの優先順位を取得
SetThreadPriority 指定したスレッドの優先順位を設定
GetCurrentProcess 現在のプロセスに対応する疑似ハンドルを取得
GetCurrentThread カレントスレッドの疑似ハンドルを取得
GetTickCount システムが起動してからの経過時間を取得

優先順位:高(1～10000カウント中割り込みできない)

優先順位:低(カウント中、フォームをクリックするとボタンクリックからの経過時間が表示される)

スレッドの優先度の指定	基本優先度
リアルタイム	24
高	13
通常以上	10
通常	8
通常以下	6
低	4

```
'=======================================
'= 指定したスレッドの優先順位を設定
'=    (SetPriorityClass.bas)
'=======================================
#include "Windows.bi"

' 指定のプロセスのプライオリティクラスを取得
Declare Function Api_GetPriorityClass& Lib "kernel32" Alias "GetPriorityClass" (ByVal
hProcess&)

' 指定のプロセスのプライオリティクラスを設定
Declare Function Api_SetPriorityClass& Lib "kernel32" Alias "SetPriorityClass" (ByVal
hProcess&, ByVal dwPriorityClass&)

' 指定したスレッドの優先順位を取得
Declare Function Api_GetThreadPriority& Lib "kernel32" Alias "GetThreadPriority" (ByVal
hThread&)

' 指定したスレッドの優先順位を設定
Declare Function Api_SetThreadPriority& Lib "kernel32" Alias "SetThreadPriority" (ByVal
hThread&, ByVal nPriority&)

' 現在のプロセスに対応する疑似ハンドルを取得
Declare Function Api_GetCurrentProcess& Lib "Kernel32" Alias "GetCurrentProcess" ()

' カレントスレッドの擬似ハンドルを取得
Declare Function Api_GetCurrentThread& Lib "kernel32" Alias "GetCurrentThread" ()

' システムが起動してからの経過時間を取得
Declare Function Api_GetTickCount& Lib "Kernel32" Alias "GetTickCount" ()

#define NORMAL_PRIORITY_CLASS &H20          '通常クラス(一般的なプロセス)
```

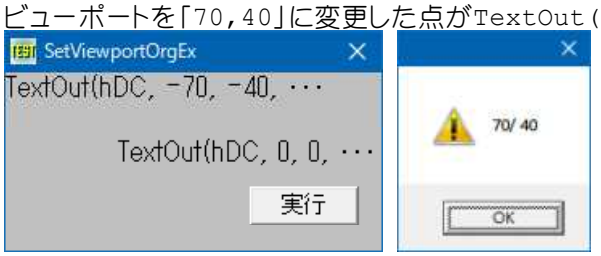
```
#define IDLE_PRIORITY_CLASS &H40            'アイドルクラス(システムがアイドル状態のときにだけスレッドを実行するプロセスであることを指定)
#define HIGH_PRIORITY_CLASS &H80            '優先クラス(すぐに実行されなければならないタスクを実行するプロセスであることを指定)
#define REALTIME_PRIORITY_CLASS &H100       'リアルタイムクラス(最優先順位クラスを持つプロセスであることを指定)
#define ABOVE_NORMAL_PRIORITY_CLASS &H8000        'NORMAL_PRIORITY_CLASS以上HIGH_PRIORITY_CLASS以下の優先度を持つ(Windows2000/XP)
#define BELOW_NORMAL_PRIORITY_CLASS &H4000        'IDLE_PRIORITY_CLASS以上NORMAL_PRIORITY_CLASS以下の優先度を持つ(Windows2000/XP)
#define THREAD_BASE_PRIORITY_IDLE -15       'デフォルト(Idle)
#define THREAD_BASE_PRIORITY_LOWRT 15       'デフォルト(Low)
#define THREAD_BASE_PRIORITY_MAX 2          'デフォルト(Max)
#define THREAD_BASE_PRIORITY_MIN -2         'デフォルト(Min)
#define THREAD_PRIORITY_HIGHEST THREAD_BASE_PRIORITY_MAX        'スレッド標準の相対優先順位値よりも2ポイント高い相対優先順位値
#define THREAD_PRIORITY_LOWEST THREAD_BASE_PRIORITY_MIN 'スレッド標準の相対優先順位値よりも2ポイント低い相対優先順位値
#define THREAD_PRIORITY_ABOVE_NORMAL (THREAD_PRIORITY_HIGHEST - 1) 'スレッド標準の相対優先順位値よりも1ポイント高い相対優先順位値
#define THREAD_PRIORITY_BELOW_NORMAL (THREAD_PRIORITY_LOWEST + 1)  'スレッド標準の相対優先順位値よりも1ポイント低い相対優先順位値
#define THREAD_PRIORITY_NORMAL 0            'スレッド標準の相対優先順位値
#define THREAD_PRIORITY_IDLE THREAD_BASE_PRIORITY_IDLE  '優先順位クラスが、IDLE_PRIORITY_CLASS、NORMAL_PRIORITY_CLASS、HIGH_PRIORITY_CLASSの場合、基本優先順位レベルが1
#define THREAD_PRIORITY_TIME_CRITICAL THREAD_BASE_PRIORITY_LOWRT      '優先順位クラスが、IDLE_PRIORITY_CLASS、NORMAL_PRIORITY_CLASS、HIGH_PRIORITY_CLASSの場合、基本優先順位レベルが15
Var Shared Picture1 As Object

Picture1.Attach GetDlgItem("Picture1") : Picture1.SetFontName "MS ゴシック"

Var Shared TimeNeed As Long
Var Shared Flg As Integer

'=====================================
'=
'=====================================
Declare Sub DoCount ()
Sub DoCount()
    Var i As Integer
    Var OldClassP As Long
    Var OldThreadP As Long
    Var Ret As Long

    OldClassP = Api_GetPriorityClass(Api_GetCurrentProcess)
    OldThreadP = Api_GetThreadPriority(Api_GetCurrentThread)

    '現在走っているプロセスの優先順位クラスを設定
    If Flg = 0 Then

        'REALTIME_PRIORITY_CLASS(優先度レベル24)
        Ret = Api_SetPriorityClass(Api_GetCurrentProcess, REALTIME_PRIORITY_CLASS)
        Ret = Api_SetThreadPriority(Api_GetCurrentThread,
THREAD_PRIORITY_TIME_CRITICAL)
    Else

        'IDLE_PRIORITY_CLASS(優先度レベル4)
        Ret = Api_SetPriorityClass(Api_GetCurrentProcess, IDLE_PRIORITY_CLASS)
        Ret = Api_SetThreadPriority(Api_GetCurrentThread, THREAD_PRIORITY_IDLE)
    End If

    TimeNeed = Api_GetTickCount
```

```
    For i = 0 To 10000
        Picture1.Cls
        Picture1.Print "Count:" & Str$(i)
        CallEvent
    Next i
    Picture1.Cls
    Picture1.Print Str$((Api_GetTickCount - TimeNeed) / 1000) & "秒"

    'プライオリティクラス、スレッドの優先順位を戻す
    Ret = Api_SetPriorityClass(Api_GetCurrentProcess, OldClassP)
    Ret = Api_SetThreadPriority(Api_GetCurrentThread, OldThreadP)
End Sub

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Flg = 0
    DoCount
End Sub

'=======================================
'=
'=======================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Flg = 1
    DoCount
End Sub

'=======================================
'=
'=======================================
Declare Sub MainForm_Click edecl ()
Sub MainForm_Click()
    A% = MessageBox("", Str$((Api_GetTickCount - TimeNeed) / 1000) & "秒", 0, 2)
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

指定したデバイスコンテキストのウィンドウ原点を指定

SetViewportOrgEx デバイスのどの点がウィンドウの原点を指しているのか指定
TextOut 文字を描画
GetDC 指定されたウィンドウのデバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放

ビューポートを「70,40」に変更した点がTextOut(0,0･･･の位置になる。

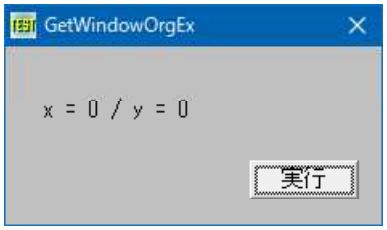


```
'=====================================
'= 指定したデバイスコンテキストのウィンドウ原点を指定
'=    (SetViewportOrgEx.bas)
'=====================================
#include "Windows.bi"

Type POINTAPI
    x   As Long
    y   As Long
End Type

#define TRANSPARENT 1                          '背景色を設定しない

' デバイスのどの点がウィンドウの原点を指しているのか指定
Declare Function Api_SetViewportOrgEx& Lib "gdi32" Alias "SetViewportOrgEx" (ByVal hDC&,
ByVal nX&, ByVal nY&, lpPoint As POINTAPI)

' 指定されたデバイスコンテキストのビューポートの原点のx座標とy座標を取得
Declare Function Api_GetViewportOrgEx& Lib "gdi32" Alias "GetViewportOrgEx" (ByVal hDC&,
lpPoint As POINTAPI)

' 文字を描画
Declare Function Api_TextOut& Lib "gdi32" Alias "TextOutA" (ByVal hDC&, ByVal nXStart&,
ByVal nYStart&, ByVal lpString$, ByVal cbString&)

' バックグラウンドの塗りつぶしモード設定
Declare Function Api_SetBkMode& Lib "gdi32" Alias "SetBkMode" (ByVal hDC&, ByVal
iBkMode&)

' 指定されたウィンドウのデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

Var Shared Button1 As Object

Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var pa As POINTAPI
    Var hDC As Long
    Var txt As String
    Var Ret As Long

    hDC = Api_GetDC(GethWnd)

    'ビューポートの原点を変更
    Ret = Api_SetViewportOrgEx(hDC, 70, 40, pa)

    Ret = Api_SetBkMode(hDC, TRANSPARENT)

    '変更した原点に文字列を描画
    txt = "TextOut(hDC, 0, 0, ···"
    Ret = Api_TextOut(hDC, 0, 0, txt, Len(txt))

    txt = "TextOut(hDC, -70, -40, ···"
    Ret = Api_TextOut(hDC, -70, -40, txt, Len(txt))

    Ret = Api_GetViewportOrgEx(hDC, pa)
    A% = MessageBox("", Str$(pa.x) & "/" & Str$(pa.y), 0, 2)

    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub
```

```
'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

指定したデバイスコンテキストのウィンドウ原点を取得

GetWindowOrgEx 指定されたデバイスコンテキストのウィンドウ原点のx座標とy座標を取得




```
'=====================================
'= 指定したデバイスコンテキストのウィンドウ原点を取得
'=    (GetWindowOrgEx.bas)
'=====================================
#include "Windows.bi"

Type POINTAPI
    x   As Long
    y   As Long
End Type

' 指定されたデバイスコンテキストのウィンドウ原点のx座標とy座標を取得
Declare Function Api_GetWindowOrgEx& Lib "gdi32" Alias "GetWindowOrgEx" (ByVal hDC&,
lpPoint As POINTAPI)

' 指定のデバイスコンテキストの論理座標基準点の座標を設定
Declare Function Api_SetWindowOrgEx& Lib "gdi32" Alias "SetWindowOrgEx" (ByVal hDC&,
ByVal nX&, ByVal nY&, lpPoint As POINTAPI)

' 指定されたウィンドウのデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

Var Shared Text1 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var pa As POINTAPI
    Var hDC As Long
    Var Ret As Long

    hDC = Api_GetDC(GethWnd)

    'ウィンドウ原点のx座標とy座標を取得
    Ret = Api_GetWindowOrgEx(hDC, pa)
```

```
    Text1.SetWindowText "x =" & Str$(pa.x) & " / y =" & Str$(pa.y)

    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

指定したファイル属性の項目を表示

特定のファイル属性の項目を表示します。
SendMessage 指定のウィンドウにメッセージを送る

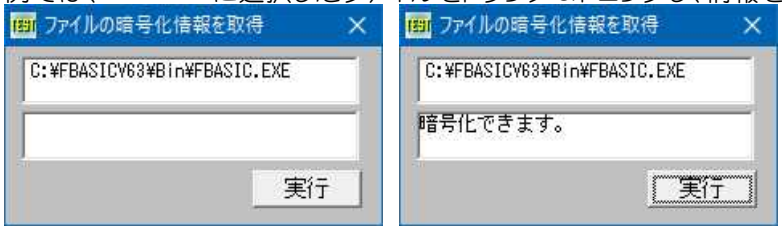

```
'=====================================
'= 指定したファイル属性の項目を表示
'=    (SendMessage7.bas)
'=====================================
#include "Windows.bi"

' ウィンドウにメッセージを送信。この関数は、指定したウィンドウのウィンドウプロシージャが処理を終了するまで制御を返さない
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal wMsg&, ByVal wParam&, lParam As Any)

#define CB_DIR &H145                    'コンボボックスにファイル名を追加
#define LB_DIR &H18D                    'リストボックスにファイル名を追加
#define DDL_ARCHIVE &H20                'アーカイブされたファイル
#define DDL_DIRECTORY &H10              '[ ]形式でディレクトリを表示
#define DDL_DRIVES &H4000               '[-x-]形式でドライブを表示
#define DDL_EXCLUSIVE &H8000            '指定された属性のファイルだけをリスト表示
#define DDL_HIDDEN &H2                  '隠しファイル
#define DDL_READONLY &H1                '読み取り専用ファイル
#define DDL_READWRITE &H0               '読み取り・書き込み可能なファイル
#define DDL_SYSTEM &H4                  'システムファイル

Var Shared Combo1 As Object
Var Shared List1 As Object
Var Shared Edit1 As Object
Var Shared Button1 As Object

Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
```

```
    Combo1.AddString "ドライブ"
    Combo1.AddString "ディレクトリ"
    Combo1.AddString "ファイル"
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Index As Long
    Var DirAttr As Long
    Var DirFileSpec As String
    Var Ret As Long

    '取得するファイルのパスを指定
    DirFileSpec = Trim$(Edit1.GetWindowtext)

    'ファイル属性を指定
    Index = Combo1.GetCursel
    If Index = -1 Then Exit Sub

    Select Case Index
        Case 0
            DirAttr = DDL_EXCLUSIVE Or DDL_DRIVES
        Case 1
            DirAttr = DDL_EXCLUSIVE Or DDL_DIRECTORY
        Case 2
            DirAttr = DDL_EXCLUSIVE Or DDL_ARCHIVE Or DDL_HIDDEN
    End Select

    'リストボックスをクリア
    List1.Resetcontent

    'リストボックスに指定したファイル属性の項目を追加
    Ret = Api_SendMessage(List1.GethWnd, LB_DIR, DirAttr, DirFileSpec)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

指定したファイルの暗号化情報を取得

FileEncryptionStatus 指定したファイルの暗号化情報を取得

例では、EditBoxに選択したファイルをドラッグ＆ドロップし、情報を取得しています。




```
'=====================================
'= 指定したファイルの暗号化情報を取得
'=    (FileEncryptionStatus.bas)
'=====================================
#include "Windows.bi"
```

```
' 指定したファイルの暗号化情報を取得
Declare Function Api_FileEncryptionStatus& Lib "advapi32" Alias "FileEncryptionStatusA"
(ByVal lpFileName$, lpStatus&)

#define FILE_ENCRYPTABLE 0                  '暗号化できる
#define FILE_IS_ENCRYPTED 1                 '暗号化されている
#define FILE_SYSTEM_ATTR 2                  'システムファイルは暗号化できない
#define FILE_ROOT_DIR 3                     'ルートディレクトリは暗号化できない
#define FILE_SYSTEM_DIR 4                   'システムディレクトリは暗号化できない
#define FILE_UNKNOWN 5                      '暗号化情報は不明
#define FILE_SYSTEM_NOT_SUPPORT 6           'ファイルシステムが暗号化をサポートしていない
#define FILE_USER_DISALLOWED 7              '暗号化情報は将来のために予約
#define FILE_READ_ONLY 8                    'ファイルは読み取り専用
#define FILE_DIR_DISALLOWED 9               'ディレクトリ暗号化抑制中は暗号化不能を示す

Var Shared Edit1 As Object
Var Shared Text1 As Object
Var Shared Button1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 12
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 12
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var Shared FileName As String

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var TargetStatus As Long
    Var Ret As Long

    'ファイルの暗号化状態を取得
    Ret = Api_FileEncryptionStatus(FileName, TargetStatus)

    ' 暗号化状態を表示
    Select Case TargetStatus
        '暗号化可能のときは
        Case FILE_ENCRYPTABLE
            Text1.SetWindowText "暗号化できます。"
        '暗号化済みのときは
        Case FILE_IS_ENCRYPTED
            Text1.SetWindowText "暗号化されています。"
        'システムファイルのときは
        Case FILE_SYSTEM_ATTR
            Text1.SetWindowText "システムファイルは" & Chr$(13, 10) & "暗号化できません。"
        'ルートディレクトリのときは
        Case FILE_ROOT_DIR
            Text1.SetWindowText "ルートディレクトリは" & Chr$(13, 10) & "暗号化できません。"
        'システムディレクトリのときは
        Case FILE_SYSTEM_DIR
            Text1.SetWindowText "システムディレクトリは" & Chr$(13, 10) & "暗号化できません。"
        '暗号化状態が不明のときは
        Case FILE_UNKNOWN
            Text1.SetWindowText "暗号化状態は不明です。"
        '暗号化非サポートのときは
        Case FILE_SYSTEM_NOT_SUPPORT
            Text1.SetWindowText "暗号化はサポートされていません。"
        '読み取り専用のときは
        Case FILE_READ_ONLY
            Text1.SetWindowText "読み取り専用は" & Chr$(13, 10) & "暗号化できません。"
        'ディレクトリ暗号化抑制中のときは
        Case FILE_DIR_DISALLOWED
            Text1.SetWindowText "暗号化抑制中は" & Chr$(13, 10) & "暗号化できません。"
        'その他のときは
        Case Else
            Text1.SetWindowText "暗号化情報は判定できません。"
    End Select
End Sub
```

```
'=====================================
'= シェルドロップされたファイル名を取得
'=====================================
Declare Sub Edit1_DropFiles edecl (ByVal DF As Long)
Sub Edit1_DropFiles(ByVal DF As Long)
    Var CN As Long

    CN = GetDropFileCount(DF)
    FileName = GetDropFileName(DF, 0)
    Edit1.SetWindowtext FileName
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

指定した矩形領域を無効領域に

指定した領域に文字を描画し、指定時間経過後その領域を無効化し終了します。
CreateDC 指定されたデバイスのデバイスコンテキストを、指定された名前で作成
DeleteDC 指定されたデバイスコンテキストを削除
DrawText 文字列を指定領域に出力
SetTextColor デバイスコンテキストの文字色を変更
WindowFromDC デバイスコンテキストからウィンドウハンドルを取得
InvalidateRect クライアント領域の一部を無効領域に
SetBkMode バックグラウンドの塗りつぶしモード設定
TRANSPARENT(1) 背景色を設定しない

指定領域(80,66)-(250,86)に「北海道札幌市白石区」という文字列を描画し、10までカウント後その領域を無効化し終了します。R・G・Bで文字色を設定します。


```
'=====================================
'= 指定した矩形領域を無効領域に
'=    (InvalidateRect.bas)
'=====================================
#include "Windows.bi"

Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

' 指定されたデバイスのデバイスコンテキストを、指定された名前で作成
Declare Function Api_CreateDC& Lib "gdi32" Alias "CreateDCA" (ByVal lpDriverName$,
lpDeviceName As Any, lpOutput As Any, ByVal lpInitData As Any)

' 指定されたデバイスコンテキストを削除
Declare Function Api_DeleteDC& Lib "gdi32" Alias "DeleteDC" (ByVal hDC&)

' 文字列を指定領域に出力
Declare Function Api_DrawText& Lib "user32" Alias "DrawTextA" (ByVal hDC&, ByVal lpStr$,
ByVal nCount&, lpRect As RECT, ByVal wFormat&)
```

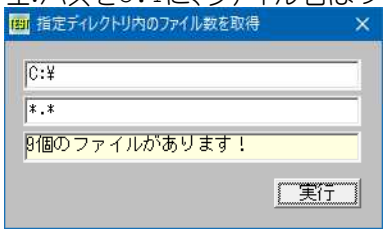
```
' デバイスコンテキストの文字色を変更
Declare Function Api_SetTextColor& Lib "gdi32" Alias "SetTextColor" (ByVal hDC&, ByVal
crColor&)

' デバイスコンテキストからウィンドウハンドルを取得
Declare Function Api_WindowFromDC& Lib "user32" Alias "WindowFromDC" (ByVal dDC&)

' クライアント領域の一部を無効領域に
Declare Function Api_InvalidateRect& Lib "user32" Alias "InvalidateRect" (ByVal hWnd&,
lpRect As RECT, ByVal bErase&)

' バックグラウンドの塗りつぶしモード設定
Declare Function Api_SetBkMode& Lib "gdi32" Alias "SetBkMode" (ByVal hDC&, ByVal
iBkMode&)

#define TRANSPARENT 1                          '背景色を設定しない

Var Shared Timer1 As Object
Var Shared Button1 As Object
Var Shared Text(2) As Object
Var Shared Edit(2) As Object

Timer1.Attach GetDlgItem("Timer1")
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
For i = 0 To 2
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1))) : Text(i).SetFontSize 14
    Edit(i).Attach GetDlgItem("Edit" & Trim$(Str$(i + 1))) : Edit(i).SetFontSize 14
Next

Var Shared count As Integer
Var Shared hWnd As Long
Var Shared rct As RECT

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Timer1.SetInterval 50
    Timer1.Enable 0
End Sub

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var hDC As Long
    Var txt As String
    Var Red As Byte
    Var Green As Byte
    Var Blue As Byte
    Var col As Long
    Var Ret As Long
    txt = "北海道札幌市白石区"

    'テキストを描画する矩形を設定
    rct.Left = 80
    rct.Top = 66
    rct.Right = 250
    rct.Bottom = 86

    'テキストカラーの数値を取得
    Red = Val(Edit(0).GetWindowText)
    Green = Val(Edit(1).GetWindowText)
    Blue = Val(Edit(2).GetWindowText)

    'デスクトップのデバイスコンテキストを取得
    hDC = Api_CreateDC("Display", ByVal 0, ByVal 0, ByVal 0)
```

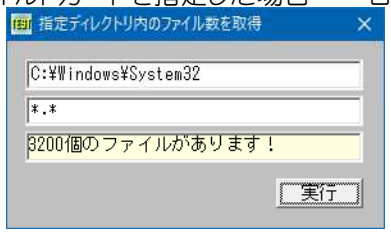
```
    'ウィンドウハンドルを取得
    hWnd = Api_WindowFromDC(hDC)

    'テキストのカラーを設定
    Ret = Api_SetTextColor(hDC, RGB(Red, Green, Blue))

    '背景色を透過
    Ret = Api_SetBkMode(hDC, TRANSPARENT)

    'テキストを描画
    Ret = Api_DrawText(hDC, txt, Len(txt), rct, 0)

    'カウント開始
    count = count + 1
    SetWindowText "InvalidateRect  " & Str$(count)

    'カウント10を経過した場合
    If count = 10 Then

        '矩形領域内のテキストを消去
        Ret = Api_InvalidateRect(hWnd, rct, 0)
        Ret = Api_DeleteDC(hDC)
        SetWindowText "InvalidateRect"
        Timer1.Enable 0
    End If
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Timer1.Enable -1
    count = 0
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
    Var Ret As Long

    Ret = Api_InvalidateRect(hWnd, rct, 0)
    Ret = Api_DeleteDC(hDC)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

指定ディレクトリ内のファイル数を取得

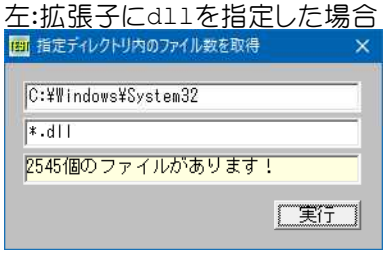
指定したディレクトリ内のファイル数を取得します。ワイルドカード、および拡張子を指定することが可能です。
FindFirstFile 指定したファイル名に一致するファイルやディレクトリを検索
FindNextFile FindFirstFile()関数で検出したファイルの次を検出
FindClose ファイル検索ハンドルをクローズ

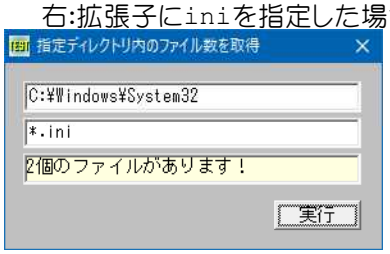
左:パスをC:¥に、ファイル名はワイルドカードを指定した場合　　右:パスをC:¥Windows¥System32に指定した場合





左:拡張子にdllを指定した場合　　右:拡張子にiniを指定した場合






```
'=====================================
'= 指定ディレクトリ内のファイル数を取得
'=    (FileCount.bas)
'=====================================
#include "Windows.bi"

#define MAX_PATH 260
#define INVALID_HANDLE_VALUE -1              '見つからない場合
#define FILE_ATTRIBUTE_DIRECTORY &H10        'ディレクトリ属性

Type FILETIME
    dwLowDateTime     As Long
    dwHighDateTime    As Long
End Type

Type WIN32_FIND_DATA
    dwFileAttributes  As Long
    ftCreationTime    As FILETIME
    ftLastAccessTime  As FILETIME
    ftLastWriteTime   As FILETIME
    nFileSizeHigh     As Long
    nFileSizeLow      As Long
    dwReserved1       As Long
    cFileName         As String * MAX_PATH
    cAlternate        As String * 14
End Type

' 指定したファイル名に一致するファイルやディレクトリを検索
Declare Function Api_FindFirstFile& Lib "kernel32" Alias "FindFirstFileA" (ByVal
lpFileName$, lpFindFileData As WIN32_FIND_DATA)

' FindFirstFile()関数で検出したファイルの次を検出
Declare Function Api_FindNextFile& Lib "kernel32" Alias "FindNextFileA" (ByVal
hFindFile&, lpFindFileData As WIN32_FIND_DATA)

' ファイル検索ハンドルをクローズ
Declare Function Api_FindClose& Lib "kernel32" Alias "FindClose" (ByVal hFindFile&)
Var Shared Edit1 As Object
Var Shared Edit2 As Object
Var Shared Text1 As Object
Var Shared Button1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Edit2.Attach GetDlgItem("Edit2") : Edit2.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
```

```
Sub MainForm_Start ()
    Edit1.SetWindowText "C:¥"
    Edit2.SetWindowText "*.*"
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var FindData As WIN32_FIND_DATA
    Var sPath As String
    Var sName As String
    Var CountFile As Integer
    Var hFile As Long

    sPath = Edit1.GetWindowText()
    sName = Edit2.GetWindowText()
    CountFile = 0

    If Right$(sPath, 1) <> "¥" Then sPath = sPath & "¥"
    sPath = sPath & sName

    hFile = Api_FindFirstFile(sPath, FindData)

    If hFile = INVALID_HANDLE_VALUE Then
        Text1.SetWindowText "該当するファイルはありません！"
        Exit Sub
    End If

    If (FindData.dwFileAttributes And FILE_ATTRIBUTE_DIRECTORY) = 0 Then CountFile = 1

    Do While Api_FindNextFile(hFile, FindData)
        If (FindData.dwFileAttributes And FILE_ATTRIBUTE_DIRECTORY) = 0 Then CountFile =
CountFile + 1
    Loop

    Text1.SetWindowText Trim$(Str$(CountFile)) & "個のファイルがあります！"

    Ret = Api_FindClose(hFile)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

指定デバイスに直接制御コードを送る

指定したデバイスドライバに直接制御コードを送ります。例では取り出し可能なドライブ名を入力し実行ボタンにより EJECTしています。
GetVersion オペレーティングシステムの種類やバージョンに関する情報を取得
CreateFile 指定したファイルをオープンし、デバイスハンドルを返す
DeviceIoControl 指定したデバイスドライバに直接制御コードを送る
CloseHandle オープンされているオブジェクトハンドルをクローズ

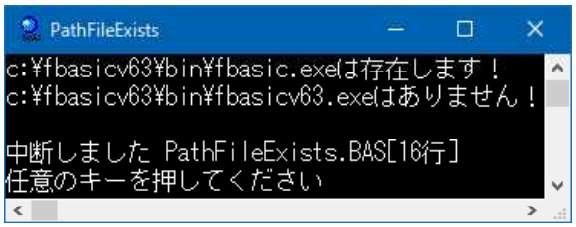
Windows98/MeとWindows2000/XPでは方法が異なるのでOSバージョンを取得し振り分けています。
GetVersion >= 0 : Windows2000
GetVersion < 0 : Windows9x


```
'=====================================
'=  指定デバイスに直接制御コードを送る
'=    (DeviceIoControl.bas)
'=====================================
#include "Windows.bi"

Type DIOC_REGISTERS
    reg_EBX    As Long
    reg_EDX    As Long
    reg_ECX    As Long
    reg_EAX    As Long
    reg_EDI    As Long
    reg_ESI    As Long
    reg_Flags  As Long
End Type

' オペレーティングシステムの種類やバージョンに関する情報を取得
Declare Function Api_GetVersion& Lib "kernel32" Alias "GetVersion" ()

' 指定したファイルをオープンし、デバイスハンドルを返す
Declare Function Api_CreateFile& Lib "kernel32" Alias "CreateFileA" (ByVal lpFileName$,
ByVal dwDesiredAccess&, ByVal dwShareMode&, lpSecurityAttributes As Any, ByVal
dwCreationDisposition&, ByVal dwFlagsAndAttributes&, ByVal hTemplateFile&)

' 指定したデバイスドライバに直接制御コードを送る
Declare Function Api_DeviceIoControl& Lib "kernel32" Alias "DeviceIoControl" (ByVal
hDevice&, ByVal dwIoControlCode&, lpInBuffer As Any, ByVal nInBufferSize&, lpOutBuffer
As Any, ByVal nOutBufferSize&, lpBytesReturned&, lpOverlapped As Any)

' オープンされているオブジェクトハンドルをクローズ
Declare Function Api_CloseHandle& Lib "kernel32" Alias "CloseHandle" (ByVal hObject&)

#define INVALID_HANDLE_VALUE -1
#define OPEN_EXISTING 3
#define FILE_FLAG_DELETE_ON_CLOSE &H4000000      '67108864
#define GENERIC_READ -2147483648                 '&H80000000
#define GENERIC_WRITE &H40000000
#define IOCTL_STORAGE_EJECT_MEDIA 2967560        'SCSIデバイスからメディアをイジェクトする
#define VWIN32_DIOC_DOS_IOCTL 1

Var Shared Edit1 As Object
Edit1.Attach getDlgItem("Edit1")

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Drive As Long
    Var Dummy As Long
    Var EjectDrive As String
    Var DriveLetter As String
    Var RawStuff As DIOC_REGISTERS
    Var Ret As Long

    EjectDrive = GetDlgItemText("Edit1")
    If Len(EjectDrive) Then

        ' 小文字なら大文字に変換し":"を付加
        DriveLetter = ucase$(Left$(EjectDrive & ":", 2))
        If Api_GetVersion >= 0 Then
```

```
            ' Windows NT･2000･XP
            Drive = Api_CreateFile("¥¥.¥" & DriveLetter, GENERIC_READ Or GENERIC_WRITE, 0,
ByVal 0, OPEN_EXISTING, 0, 0)
            If Drive <> INVALID_HANDLE_VALUE Then

                ' メディアイジェクト
                Ret = Api_DeviceIoControl(Drive, IOCTL_STORAGE_EJECT_MEDIA, 0, 0, 0, 0,
Dummy, ByVal 0)
                Ret = APi_CloseHandle(Drive)
            End If
        Else

            ' Win9x･Me
            Drive = Api_CreateFile("¥¥.¥VWIN32", 0, 0, ByVal 0, 0,
FILE_FLAG_DELETE_ON_CLOSE, 0)
            If Drive <> INVALID_HANDLE_VALUE Then

                ' 仮想デバイスドライバを通してInt21h Function 440Dh Minor Code 49hを実行
                RawStuff.reg_EAX = &H440D
                RawStuff.reg_EBX = asc(DriveLetter) - Asc("A") + 1
                RawStuff.reg_ECX = &H49 Or &H800

                ' メディアイジェクト
                Ret = Api_DeviceIoControl(Drive, VWIN32_DIOC_DOS_IOCTL, RawStuff,
Len(RawStuff), RawStuff, Len(RawStuff), Dummy, ByVal 0)
                Ret = Api_CloseHandle(Drive)
            End If
        End If
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

指定のファイルの有無をチェック

指定したファイルが存在するかどうかを判定します。
PathFileExists 指定のファイルの有無をチェック



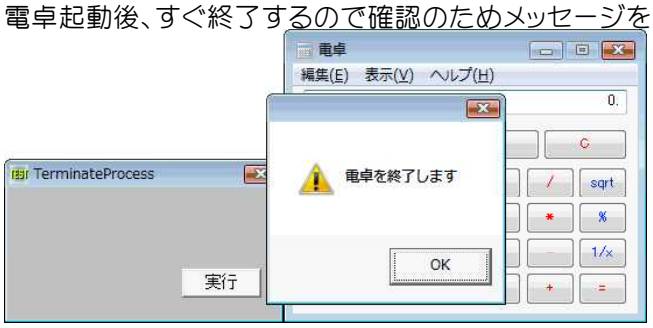

```
'=====================================
'= ファイルの有無をチェック
'=    (PathFindExists.bas/P)
'=====================================

' 指定のファイルの有無をチェック
Declare Function Api_PathFileExists& Lib "shlwapi" Alias "PathFileExistsA" (ByVal
lpszPath$)

Var FileName As String
Var Ret As Long

FileName = "c:¥fbasicv63¥bin¥fbasic.exe"
```

```
    GoSub *judge
FileName = "c:¥fbasicv63¥bin¥fbasicv63.exe"
    GoSub *judge

Stop
End

'-------------------------------------------------------------------
*judge
Ret = Api_PathFileExists(FileName)

If Ret <> 0 Then
    Print FileName & "は存在します！"
Else
    Print FileName & "はありません！"
End If
Return
```

指定のプロセスを強制終了させる

電卓を起動し、強制終了させています。
CreateProcess プロセスの起動
TerminateProcess 指定のプロセスを強制終了させる
CloseHandle オブジェクトハンドルをクローズ

電卓起動後、すぐ終了するので確認のためメッセージを表示させ一旦停止させています。




```
'====================================
'= 指定のプロセスを強制終了させる
'=    (TerminateProcess.bas)
'====================================
#include "Windows.bi"

Type PROCESS_INFORMATION
    hProcess          As Long
    hThread           As Long
    dwProcessId       As Long
    dwThreadId        As Long
End Type

Type STARTUPINFO
    cb                As Long
    lpReserved        As Long
    lpDesktop         As Long
    lpTitle           As LOng
    dwX               As Long
    dwY               As Long
    dwXSize           As Long
    dwYSize           As Long
    dwXCountChars     As Long
    dwYCountChars     As Long
    dwFillAttribute   As Long
    dwFlags           As Long
    wShowWindow       As Integer
    cbReserved2       As Integer
    lpReserved2       As Long
```

```
    hStdInput         As Long
    hStdOutput        As Long
    hStdError         As Long
End Type

' プロセスの起動
Declare Function Api_CreateProcess& Lib "kernel32" Alias "CreateProcessA" (ByVal
lpName&, ByVal lpComLine$, lpPAttr As Any, lpTAttr As Any, ByVal bHndl&, ByVal dwCFlg&,
lpEnv As Any, ByVal lpCDir$, lpSInfo As STARTUPINFO, lpProc As PROCESS_INFORMATION)

' 指定のプロセスを強制終了させる
Declare Function Api_TerminateProcess& Lib "kernel32" Alias "TerminateProcess" (ByVal
hProcess&, ByVal uExitCode&)

' オープンされているオブジェクトハンドルをクローズ
Declare Function Api_CloseHandle& Lib "Kernel32" Alias "CloseHandle" (ByVal hObject&)

#define SYNCHRONIZE &H100000
#define PROCESS_TERMINATE &H1
#define NORMAL_PRIORITY_CLASS &H20

Var Shared Button1 As Object

Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'====================================
'=
'====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    ShowWindow -1
    Cls
End Sub

'====================================
'=
'====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var pi As PROCESS_INFORMATION
    Var StartInfo As STARTUPINFO
    Var ProcHandle As Long
    Var Ret As Long

    StartInfo.cb = Len(StartInfo)
    Ret = Api_CreateProcess(0, "Calc.exe", ByVal 0, ByVal 0, 1, NORMAL_PRIORITY_CLASS,
ByVal 0, ByVal 0, StartInfo, pi)

    '確認のため一旦制御を止める
    A% = MessageBox("", "電卓を終了します", 0, 2)

    '電卓を終了
    Ret = Api_TerminateProcess(pi.hProcess, 0)

    'ハンドルの解放
    Ret = Api_CloseHandle(pi.hThread)
    Ret = Api_CloseHandle(pi.hProcess)
End Sub

'====================================
'=
'====================================
While 1
    WaitEvent
Wend
Stop
End
```

指定のメニュー項目にチェックを入れる

指定したメニュー項目にチェックを入れます。
GetMenu メニューハンドルを取得
GetSubMenu ポップアップメニューのハンドルを取得
CheckMenuRadioItem 指定範囲のメニュー項目をグループ化

例では、0～1、3～6をそれぞれグループ化しています。


```
'========================================
'= 指定のメニュー項目にチェックを入れる
'=    (CheckMenuRadioItem.bas)
'========================================
#include "Windows.bi"

' メニューのハンドルを取得
Declare Function Api_GetMenu& Lib "user32" Alias "GetMenu" (ByVal hWnd&)

' ポップアップメニューのハンドルを取得
Declare Function Api_GetSubMenu& Lib "user32" Alias "GetSubMenu" (ByVal hMenu&, ByVal
nPos&)

' 指定の範囲のメニュー項目をグループ化
Declare Function Api_CheckMenuRadioItem& Lib "user32" Alias "CheckMenuRadioItem" (ByVal
hMenu&, ByVal idFirst&, ByVal idLast&, ByVal idCheck&, ByVal uFlag&)

#define MF_APPEND &H100                    '
#define MF_BYCOMMAND &H0                   'nPositionはメニュー項目のID
#define MF_BYPOSITION &H400                'nPositionはメニュー項目のインデックス
#define MF_CHECKED &H8                     'メニュー項目にチェックをつける
#define MF_DISABLED &H2                    'アイテムを選択不可にする
#define MF_GRAYED &H1                      'グレー表示されて選択できない
#define MF_HILITE &H80                     'メニューの項目を強調表示
#define MF_MOVE &H1000                     '
#define MF_SEPARATOR &H800                 'メニュー項目はセパレータ
#define MF_STRING &H0                      '文字列
#define MF_UNHILITE &H0                    'メニューの項目を強調表示しない

Var Shared SubMenu As Long

'========================================
'=
'========================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var Menu As Long
    Var hWnd
    Var Ret As Long

    hWnd = GethWnd

    Menu = Api_GetMenu(hWnd)

    SubMenu = Api_GetSubMenu(Menu, 0)
End Sub

'========================================
'=
'========================================
```

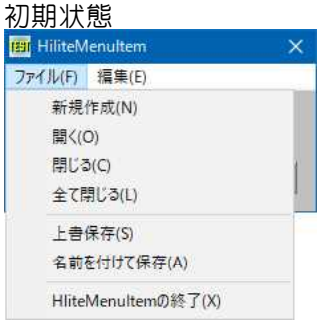
```
Declare Sub mnuItem1_on edecl ()
Sub mnuItem1_on()
    Var Ret As Long

    Ret = Api_CheckMenuRadioItem(SubMenu, 0, 1, 0, MF_BYPOSITION)
End Sub

'=====================================
'=
'=====================================
Declare Sub mnuItem2_on edecl ()
Sub mnuItem2_on()
    Var Ret As Long

    Ret = Api_CheckMenuRadioItem(SubMenu, 0, 1, 1, MF_BYPOSITION)
End Sub

'=====================================
'=
'=====================================
Declare Sub mnuItem3_on edecl ()
Sub mnuItem3_on()
    Var Ret As Long

    Ret = Api_CheckMenuRadioItem(SubMenu, 3, 6, 3, MF_BYPOSITION)
End Sub

'=====================================
'=
'=====================================
Declare Sub mnuItem4_on edecl ()
Sub mnuItem4_on()
    Var Ret As Long

    Ret = Api_CheckMenuRadioItem(SubMenu, 3, 6, 4, MF_BYPOSITION)
End Sub

'=====================================
'=
'=====================================
Declare Sub mnuItem5_on edecl ()
Sub mnuItem5_on()
    Var Ret As Long

    Ret = Api_CheckMenuRadioItem(SubMenu, 3, 6, 5, MF_BYPOSITION)
End Sub

'=====================================
'=
'=====================================
Declare Sub mnuItem6_on edecl ()
Sub mnuItem6_on()
    Var Ret As Long

    Ret = Api_CheckMenuRadioItem(SubMenu, 3, 6, 6, MF_BYPOSITION)
End Sub

'=====================================
'=
'=====================================
Declare Sub mnuExit_on edecl ()
Sub mnuExit_on()
    End
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
```

```
Wend
Stop
End
```

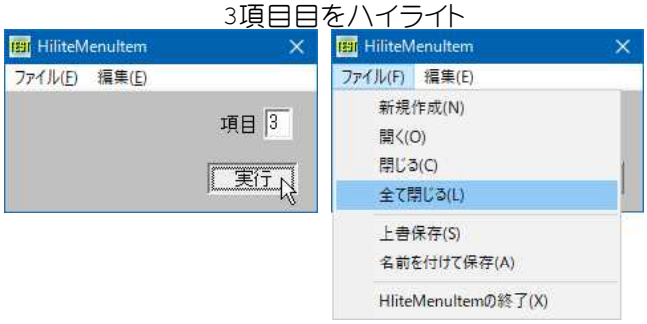
指定のメニュー項目を強調表示（ハイライト）

メニューの項目番号を指定し、その文字列を強調表示させます。
GetMenu メニューのハンドルを取得
GetSubMenu ポップアップメニューのハンドルを取得
HiliteMenuItem 指定のメニュー項目を強調表示

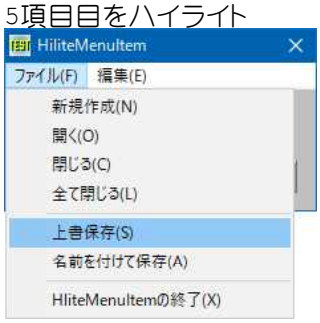
初期状態



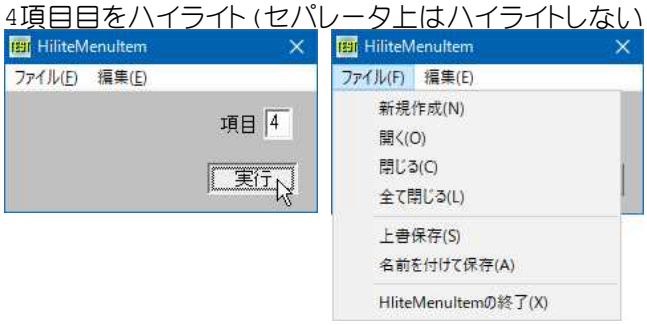
3項目目をハイライト



5項目目をハイライト



4項目目をハイライト（セパレータ上はハイライトしない）




```
'========================================
'= 指定のメニュー項目を強調表示（ハイライト）
'=    (HiliteMeniItem.bas)
'========================================
#include "Windows.bi"

' メニューのハンドルを取得
Declare Function Api_GetMenu& Lib "user32" Alias "GetMenu" (ByVal hWnd&)

' ポップアップメニューのハンドルを取得
Declare Function Api_GetSubMenu& Lib "user32" Alias "GetSubMenu" (ByVal hMenu&, ByVal
nPos&)

' 指定の項目を強調表示
Declare Function Api_HiliteMenuItem& Lib "user32" Alias "HiliteMenuItem" (ByVal hWnd&,
ByVal hMenu&, ByVal wIDHiliteItem&, ByVal wHilite&)

#define MF_APPEND &H100                    '
#define MF_BYCOMMAND &H0                   'nPositionはメニュー項目のID
#define MF_BYPOSITION &H400                'nPositionはメニュー項目のインデックス
#define MF_CHECKED &H8                     'メニュー項目にチェックをつける
#define MF_DISABLED &H2                    'アイテムを選択不可にする
#define MF_GRAYED &H1                      'グレー表示されて選択できない
#define MF_HILITE &H80                     'メニューの項目を強調表示
#define MF_MOVE &H1000                     '
#define MF_SEPARATOR &H800                 'メニュー項目はセパレータ
#define MF_STRING &H0                      '文字列
#define MF_UNHILITE &H0                    'メニューの項目を強調表示しない

Var Shared Edit1 As Object
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
```

```
'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var MenuIdx As Long
    Var hMenu As Long
    Var SubMenu As Long
    Var Ret As Long

    MenuIdx = Val(Edit1.GetWindowText)
    hMenu = Api_GetMenu(GethWnd)
    SubMenu = Api_GetSubMenu(hMenu, 0)
    Ret = Api_HiliteMenuItem(GethWnd, SubMenu, MenuIdx, MF_BYPOSITION Or MF_HILITE)
    refresh
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

指定のメニュー項目を強調表示(太字)

指定したメニュー項目を強調(太字)します。
GetMenu メニューのハンドルを取得
GetSubMenu サブメニューを開く場合そのメニューのハンドルを取得
SetMenuItemInfo メニュー項目に関する情報を変更
DrawMenuBar 指定されたウィンドウのメニューバーを再描画

例では、ヘルプ(H)、ページ設定(U)、コピー(C)を強調(太字)設定しています。
メニューバーに対して何らかの変更を加えた場合、その直後にDrawMenuBarでメニューバーを再描画させます。
引数はそのメニューバーが属するウィンドウのハンドルを指定します。

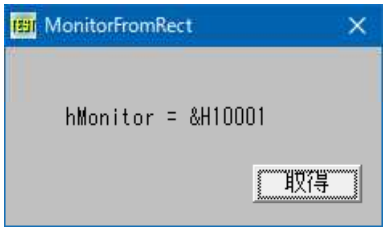

```
'=====================================
'= 指定のメニュー項目を強調表示(太字)
'=    (SetmenuItemInfo2.bas)
'=====================================
#include "Windows.bi"

#define API_FALSE 0
#define API_TRUE 1

#define MFT_BITMAP &H4                  'ビットマップを使ってメニュー項目を表示
#define MFT_MENUBARBREAK &H20           'メニュー項目を新しい列に置く(ドロップダウンメニュー、サブメニュー、ショートカットメニューの時は区切・
#define MFT_MENUBREAK &H40              'メニュー項目を新しい列に置く(区切り線は入らない)
#define MFT_OWNERDRAW &H100             'オーナードローをする時指定
#define MFT_RADIOCHECK &H200            'hbmpCheckedメンバがNULLの時ラジオボタンを使ってチェックされる
#define MFT_RIGHTJUSTIFY &H4000         'メニュー項目が右揃え
#define MFT_RIGHTORDER &H2000           'アラビア語とかヘブライ語の時指定(Win9x・2000)
#define MFT_SEPARATOR &H800             '区切り線(dwTypeDataとcchメンバは無視)
```

```
#define MFT_STRING &H0                          'メニュー項目に文字列を使う

#define MFS_CHECKED &H8                         'メニュー項目にチェックをつける
#define MFS_DEFAULT &H1000                      'メニュー項目がデフォルトであることを指定
#define MFS_DISABLED &H2                        'メニュー項目を選択不可にする
#define MFS_ENABLED &H0                         'メニュー項目を選択可能にする
#define MFS_GRAYED &H1                          'メニュー項目を灰色にして選択不可にする
#define MFS_HILITE &H80                         'メニュー項目をハイライトにする
#define MFS_UNCHECKED &H0                       'メニュー項目からチェックをはずす
#define MFS_UNHILITE &H0                        'メニュー項目のハイライトをやめる

#define MIIM_CHECKMARKS &H8                     'hbmpChecked、hbmpUncheckedメンバをセット
#define MIIM_DATA &H20                          'dwItemDataメンバをセット
#define MIIM_ID &H2                             'wIDメンバをセット
#define MIIM_STATE &H1                          'fStateメンバをセット
#define MIIM_SUBMENU &H4                        'hSubMenuメンバをセット
#define MIIM_TYPE &H10                          'fTypeメンバをセット

Type MENUITEMINFO
    cbSize              As Long                 '構造体のバイト数
    fMask               As Long                 '取得する情報を指定する定数の組み合わせ
    fType               As Long                 'メニュー項目のタイプを指定する定数の組み合わせ
    fState              As Long                 'メニューの状態を指定する定数の組み合わせ
    wID                 As Long                 'ユーザー定義のメニュー項目のID
    hSubMenu            As Long                 '指定のメニュー項目と関連するサブメニューのハンドル
    hbmpChecked         As Long                 'チェックマーク用のビットマップのハンドル
    hbmpUnchecked       As Long                 '未チェック時のときのビットマップハンドル
    dwItemData          As Long                 'メニュー項目と関連するユーザー定義の値
    dwTypeData          As Long                 'メニュー項目のタイプ(fMaskにMIIM_Typeを指定したときのみ有効)

    cch                 As Long                 'メニュー項目のテキストのバイト数
End Type

' 指定されたウィンドウに割り当てられているメニューのハンドルを取得
Declare Function Api_GetMenu& Lib "user32" Alias "GetMenu" (ByVal hWnd&)

' 指定されたメニュー項目がドロップダウンメニューまたはサブメニューを開く場合、そのメニューのハンドルを取得
Declare Function Api_GetSubMenu& Lib "user32" Alias "GetSubMenu" (ByVal hMenu&, ByVal
nPos&)

' メニュー項目に関する情報を変更
Declare Function Api_SetMenuItemInfo& Lib "user32" Alias "SetMenuItemInfoA" (ByVal
hMenu&, ByVal uItem&, ByVal fByPosition&, lpMInfo As MENUITEMINFO)

' 指定されたウィンドウのメニューバーを再描画
Declare Function Api_DrawMenuBar& Lib "user32" Alias "DrawMenuBar" (ByVal hWnd&)

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var hMenu As Long
    Var hSubMenu As Long
    Var mii As MENUITEMINFO

    hMenu = Api_GetMenu(GethWnd)

    If hMenu Then

        '3番目のメニュー(0が起点:ヘルプ)
        mii.cbSize = Len(mii)
        mii.fMask = MIIM_STATE
        mii.fState = MFS_DEFAULT

        Ret = Api_SetMenuItemInfo(hMenu, 2, API_TRUE, mii)

        'ファイルのサブメニューポジション
        hSubMenu = Api_GetSubMenu(hMenu, 0)
```

```
        If hSubMenu Then
            mii.cbSize = Len(mii)
            mii.fMask = MIIM_STATE
            mii.fState = MFS_DEFAULT

            '5番目のサブメニューを強調
            Ret = Api_SetMenuItemInfo(hSubMenu, 4, API_TRUE, mii)
        End If

        '編集のサブメニューポジション
        hSubMenu = Api_GetSubMenu(hMenu, 1)

        If hSubMenu Then
            mii.cbSize = Len(mii)
            mii.fMask = MIIM_STATE
            mii.fState = MFS_DEFAULT

            '2番目のサブメニューを強調
            Ret = Api_SetMenuItemInfo(hSubMenu, 1, API_TRUE, mii)
        End If
    End If
    Ret = Api_DrawMenuBar(GethWnd)
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

指定領域との交差部分が最も広いモニタのハンドルを取得

MonitorFromRect 指定された四角形との交差部分が最も大きいディスプレイモニタへのハンドルを取得

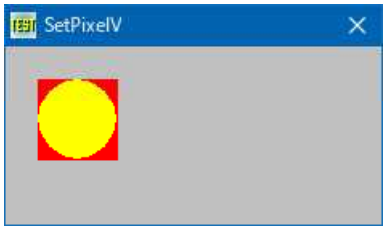



```
'=======================================
'= 指定長方形領域との交差部分が最も広いモニタのハンドルを取得
'=    (MonitorFromRect.bas)
'=======================================
#include "Windows.bi"

Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

#define MONITOR_DEFAULTTONEAREST &H2        '指定したウィンドウに最も近い位置にあるディスプレイモニタのハンドルが返る
#define MONITOR_DEFAULTTONULL &H0           'NULLが返る
#define MONITOR_DEFAULTTOPRIMARY &H1        'プライマリディスプレイモニタのハンドルが返る

' 指定された四角形との交差部分が最も大きいディスプレイモニタへのハンドルを取得
Declare Function Api_MonitorFromRect& Lib "user32" Alias "MonitorFromRect" (ByRef lprc
As RECT, ByVal dwFlags&)
```

```
Var Shared Text1 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

ShowWindow -1

'=====================================
'=
'=====================================
Declare Function GetMonitorByRect(rc As RECT) As Long
Function GetMonitorByRect(rc As RECT) As Long
    GetMonitorByRect = Api_MonitorFromRect(rc, MONITOR_DEFAULTTONEAREST)
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var rc As RECT

    rc.Left = 650
    rc.Right = 1243
    rc.Top = 455
    rc.Bottom = 950
    hMonitor = GetMonitorByRect(rc)

    Text1.SetWindowText "hMonitor = &&H" & Hex$(hMonitor)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

指定領域内をピクセルで描画

指定された矩形領域内のおよび円形領域内にポイントがあれば、ピクセルで描画します。
例では、矩形領域内を赤で、円形領域内を黄色で描画しています。
PtInRect ある点が指定の矩形領域内にあるかどうかを判定
PtInRegion 指定された点がリージョン内にあるかどうか判定
CreateEllipticRgnIndirect 円形・楕円形の領域をRECT構造体に基づいて作成
SetPixelV 指定の位置のピクセルを指定のカラーに最も近いカラー値に設定
SetRect RECT構造体の値を設定
DeleteObject オブジェクトに関連付けられていた全てのシステムリソースを解放



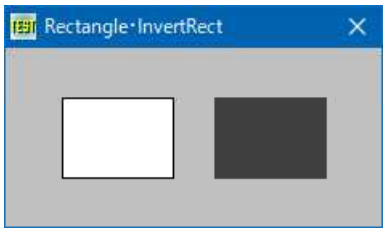

```
'=====================================
'= 指定領域内をピクセルで描画
'=    (PointIn.bas)
'=====================================
#include "Windows.bi"
```

```
Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

' オブジェクトに関連付けられていた全てのシステムリソースを解放
Declare Function Api_DeleteObject& Lib "gdi32" Alias "DeleteObject" (ByVal hObject&)

' ある点が指定の矩形領域内にあるかどうかを判定
Declare Function Api_PtInRect& Lib "user32" Alias "PtInRect" (lpRect As RECT, ByVal x&,
ByVal y&)

' 指定された点がリージョン内にあるかどうか判定
Declare Function Api_PtInRegion& Lib "gdi32" Alias "PtInRegion" (ByVal hRgn&, ByVal x&,
ByVal y&)

' 円形・楕円形の領域をRECT構造体に基づいて作成
Declare Function Api_CreateEllipticRgnIndirect& Lib "gdi32" Alias
"CreateEllipticRgnIndirect" (lpRect As RECT)

' 指定の位置のピクセルを指定のカラーに最も近いカラー値に設定
Declare Function Api_SetPixelV& Lib "gdi32" Alias "SetPixelV" (ByVal hDC&, ByVal x&,
ByVal y&, ByVal crColor&)

' RECT構造体の値を設定
Declare Function Api_SetRect& Lib "user32" Alias "SetRect" (lpRect As RECT, ByVal X1&,
ByVal Y1&, ByVal X2&, ByVal Y2&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var hDC As Long
    Var mRGN As Long
    Var R As RECT
    Var x As Long
    Var y As Long
    Var Ret As Long

    '矩形領域を設定      L   T   R   B
    Ret = Api_SetRect(R, 20, 20, 70, 70)

    'フォームのDCを取得
    hDC = Api_GetDC(GetHwnd)

    '円・楕円領域をRECT構造体に基づき作成
    mRGN = Api_CreateEllipticRgnIndirect(R)

    '領域内ポイントをスキャン
    For x = R.Left To R.Right
        For y = R.Top To R.Bottom

            'ポイントが領域内であれば黄色のピクセルで描画
            If Api_PtInRegion(mRGN, x, y) <> 0 Then
                Ret = Api_SetPixelV(hDC, x, y, rgb(255, 255, 0))

            '赤でピクセルを描画
            Else If Api_PtInRect(R, x, y) <> 0 Then
                Ret = Api_SetPixelV(hDC, x, y, rgb(255, 0, 0))
            End If
        Next y
```

```
    Next x

    Ret = Api_DeleteObject(mRGN)
    Ret = Api_ReleaseDC(GetWnd, hDC)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

指定領域に長方形を描画・反転描画

指定領域に長方形を描画(Rectangle)・反転描画(InvertRect)します。
Rectangle 長方形の描画
InvertRect 長方形内の色を反転描画



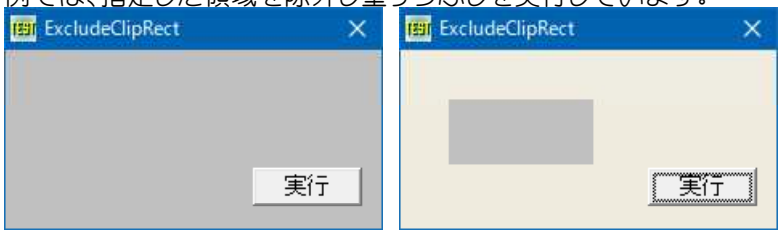

```
'=====================================
'= 指定領域に長方形を描画・反転描画
'=    (InvertRect.bas)
'=====================================
#include "Windows.bi"

Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

' 長方形の描画
Declare Function Api_Rectangle& Lib "gdi32" Alias "Rectangle" (ByVal hDC&, ByVal X1&,
ByVal Y1&, ByVal X2&, ByVal Y2&)

' 長方形内の色を反転描画
Declare Function Api_InvertRect& Lib "user32" Alias "InvertRect" (ByVal hDC&, lpRect As
RECT)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var R As RECT
    Var hDC As Long
    Var Ret As Long

    hDC = Api_GetDC(GethWnd)
```

```
    R.Left = 35
    R.Right = 105
    R.Top = 30
    R.Bottom = 80

    'Rectangle
    Ret = Api_Rectangle(hDC, R.Left, R.Top, R.Right, R.Bottom)

    R.Left = 130
    R.Right = 200

    'Invert Rectangle(InvertRect)
    Ret = Api_InvertRect(hDC, R)

    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

指定領域を除外する(ExcludeClipRect)

ExcludeClipRect 既存のクリッピング領域から、指定された長方形の領域を取り除く
ExtFloodFill 現在選択されているブラシで一定の範囲内を塗りつぶす
CreateSolidBrush ソリッドカラーで論理ブラシを作成
SelectObject 指定されたデバイスコンテキストのオブジェクトを選択
DeleteObject システムリソースを解放
GetDC デバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放

例では、指定した領域を除外し塗りつぶしを実行しています。




```
'=====================================
'= 指定領域を除外する
'=    (ExcludeClipRect.bas)
'=====================================
#include "Windows.bi"

' 既存のクリッピング領域から、指定された長方形の領域を取り除く
Declare Function Api_ExcludeClipRect& Lib "gdi32" Alias "ExcludeClipRect" (ByVal hDC&,
ByVal X1&, ByVal Y1&, ByVal X2&, ByVal Y2&)

' 現在選択されているブラシで一定の範囲内を塗りつぶす
Declare Function Api_ExtFloodFill& Lib "gdi32" Alias "ExtFloodFill" (ByVal hDC&, ByVal
X&, ByVal Y&, ByVal crColor&, ByVal wFillType&)

' ソリッドカラーで論理ブラシを作成
Declare Function Api_CreateSolidBrush& Lib "gdi32" Alias "CreateSolidBrush" (ByVal
crColor&)

' 指定されたデバイスコンテキストのオブジェクトを選択
Declare Function Api_SelectObject& Lib "gdi32" Alias "SelectObject" (ByVal hDC&, ByVal
hObject&)
```

```
' ペン、ブラシ、フォント、ビットマップ、リージョン、パレットのいずれかの論理オブジェクトを削除し、そのオブジェクトに関連付けられていたすべてのシステムリソースを解放。オブジェクトを削除した後は、指定されたハンドルは無効になる
Declare Function Api_DeleteObject& Lib "gdi32" Alias "DeleteObject" (ByVal hObject&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

#define FLOODFILLBORDER 0                    'crColorで指定した色が囲んでいる領域を、塗りつぶす
#define FLOODFILLSURFACE 1                   'crColorで指定した色と同じ色になっている領域を、塗りつぶす
'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hDC As Long
    Var hBrush As Long
    Var hOldBrush As Long
    Var Ret As Long

    hDC = Api_GetDC(GethWnd)
    hBrush = Api_CreateSolidBrush(RGB(239, 235, 222))
    hOldBrush = Api_SelectObject(hDC, hBrush)

    'クリッピング領域から除く
    Ret = Api_ExcludeClipRect(hDC, 30, 30, 120, 70)
    Ret = Api_ExtFloodFill(hDC, 1, 1, 0, FLOODFILLBORDER)

    Ret = Api_DeleteObject(hBrush)
    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

指定領域をフラッシュ

例では、指定した矩形領域をフラッシュ(反転)させています。
GdiFlush 呼び出し側スレッドの現在のバッチをフラッシュ
InvertRect 長方形内の色を反転描画
Sleep カレントスレッドの実行を指定の時間だけ中断
GetDC 指定されたウィンドウのクライアント領域のデバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放

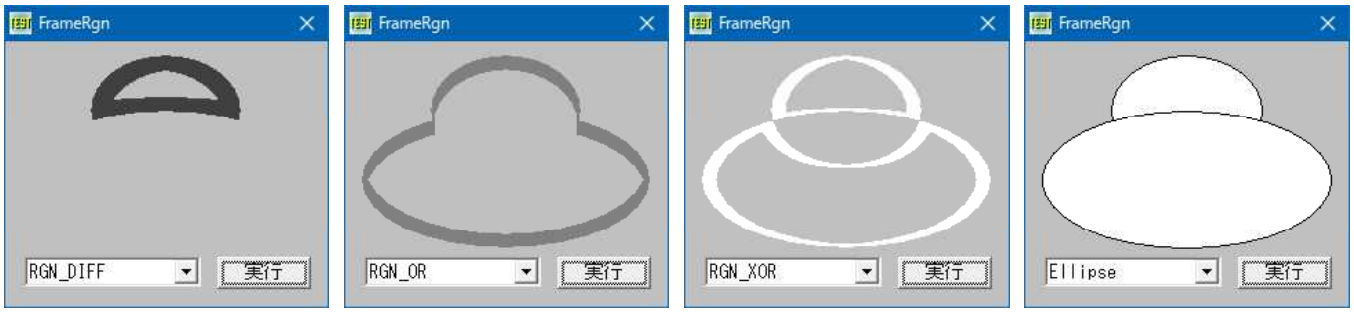

```
'=====================================
'= 指定領域をフラッシュ
'=    (GdiFlush.bas)
'=====================================
#include "Windows.bi"
```

```
Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

' 呼び出し側スレッドの現在のバッチをフラッシュ
Declare Function Api_GdiFlush& Lib "gdi32" Alias "GdiFlush" ()

' 長方形内の色を反転描画
Declare Function Api_InvertRect& Lib "user32" Alias "InvertRect" (ByVal hDC&, lpRect As
RECT)

' カレントスレッドの実行を指定の時間だけ中断
Declare Sub Api_Sleep Lib "Kernel32" Alias "Sleep" (ByVal dwMilliseconds&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

Var Shared Picture1 As Object
Var Shared Button1 As Object
Var Shared Bitmap As Object

Picture1.Attach GetDlgItem("Picture1")
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
BitmapObject Bitmap

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Bitmap.LoadFile "flower.bmp"
    Picture1.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var rct As RECT
    Var hDC As Long
    Var i As Long
    Var Ret As Long
    hDC = Api_GetDC(Picture1.GethWnd)

    rct.Left = Picture1.GetWidth / 5
    rct.Top = Picture1.GetHeight / 5
    rct.Right = Picture1.GetWidth - Picture1.GetWidth / 5
    rct.Bottom = Picture1.GetHeight - Picture1.GetHeight / 5

    For i = 1 To 10
        Ret = Api_InvertRect(hDC, rct)
        Ret = Api_GdiFlush
        Api_Sleep(200)
    Next

    Ret = Api_ReleaseDC(Picture1.GethWnd, hDC)
End Sub

'=====================================
'=
'=====================================
While 1
```

```
    WaitEvent
Wend
Stop
End
```

指定領域の周囲にブラシで境界線を描画

CreateEllipticRgn 楕円形のリージョンを作成
Ellipse 楕円の描画
CombineRgn 既存の二つの領域を結合して新しい領域を作成
FrameRgn 指定の領域の周囲に指定のブラシで境界線を描く
GetDC デバイスコンテキストのハンドルを取得
GetStockObject ストックオブジェクトのハンドルを取得
ReleaseDC デバイスコンテキストを解放
DeleteObject システムリソースを解放




```
'=====================================
'= 指定領域の周囲にブラシで境界線を描画
'=    (FrameRgn.bas)
'=====================================
#include "Windows.bi"

' 楕円形のリージョンを作成
Declare Function Api_CreateEllipticRgn& Lib "gdi32" Alias "CreateEllipticRgn" (ByVal
nLeftRect&, ByVal nTopRect&, ByVal nRightRect&, ByVal nBottomRect&)

' 楕円の描画
Declare Function Api_Ellipse& Lib "gdi32" Alias "Ellipse" (ByVal hDC&, ByVal X1&, ByVal
Y1&, ByVal X2&, ByVal Y2&)

' 既存の二つの領域を結合して新しい領域を作成
Declare Function Api_CombineRgn& Lib "gdi32" Alias "CombineRgn" (ByVal hRgnDest&, ByVal
hRgnSrc1&, ByVal hRgnSrc2&, ByVal nCombineMode&)

' 指定の領域の周囲に指定のブラシで境界線を描く
Declare Function Api_FrameRgn& Lib "gdi32" Alias "FrameRgn" (ByVal hDC&, ByVal hRgn&,
ByVal hBrush&, ByVal nWidth&, ByVal nHeight&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' ストックオブジェクトのハンドルを取得
Declare Function Api_GetStockObject& Lib "gdi32" Alias "GetStockObject" (ByVal nIndex&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

' 論理オブジェクトを削除し、そのオブジェクトに関連付けられていたすべてのシステムリソースを解放
Declare Function Api_DeleteObject& Lib "gdi32" Alias "DeleteObject" (ByVal hObject&)
#define RGN_COPY 5                          'HRGNSRC1のコピーを作成
#define RGN_DIFF 4                          'HRGNSRC1からHRGNSRC2を除いた領域
#define RGN_OR 2                            'リージョン同士のOR結合
#define RGN_XOR 3                           'リージョン同士のXOR結合
#define BLACK_BRUSH 4                       '黒
#define DKGRAY_BRUSH 3                      '暗い灰色
```

```
#define GRAY_BRUSH 2                        '灰色
#define LTGRAY_BRUSH 1                      '明るい灰色
#define WHITE_BRUSH 0                       '白

Var Shared Picture1 As Object
Var Shared Combo1 As Object
Var Shared Button1 As Object

Picture1.Attach GetDlgItem("Picture1") : Picture1.SetFontSize 14
Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Combo1.AddString "RGN_DIFF"
    Combo1.AddString "RGN_OR"
    Combo1.AddString "RGN_XOR"
    Combo1.AddString "Ellipse"
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hDC As Long
    Var hRgn1 As Long
    Var hRgn2 As Long
    Var Ret As Long

    hDC = Api_GetDC(Picture1.GethWnd)

    'リージョンを作成
    hRgn1 = Api_CreateEllipticRgn(50, 0, 160, 80)
    hRgn2 = Api_CreateEllipticRgn(0, 40, 210, 140)

    Picture1.Cls

    'リージョンを結合・楕円描画
    Select Case Combo1.GetCursel
        Case 0
            Ret = Api_CombineRgn(hRgn1, hRgn1, hRgn2, RGN_DIFF)
            Ret = Api_FrameRgn(hDC, hRgn1, Api_GetStockObject(DKGRAY_BRUSH), 10, 10)
        Case 1
            Ret = Api_CombineRgn(hRgn1, hRgn1, hRgn2, RGN_OR)
            Ret = Api_FrameRgn(hDC, hRgn1, Api_GetStockObject(GRAY_BRUSH), 2, 10)
        Case 2
            Ret = Api_CombineRgn(hRgn1, hRgn1, hRgn2, RGN_XOR)
            Ret = Api_FrameRgn(hDC, hRgn1, Api_GetStockObject(WHITE_BRUSH), 10, 2)
        Case Else
            Ret = Api_Ellipse(hDC, 50, 0, 160, 80)
            Ret = Api_Ellipse(hDC, 0, 40, 210, 140)
    End Select

    '後処理
    Ret = Api_DeleteObject(hRgn1)
    Ret = Api_DeleteObject(hRgn2)
    Ret = Api_ReleaseDC(Picture1.GethWnd, hDC)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop : End
```

自フォーム前後のウィンドウハンドルを取得

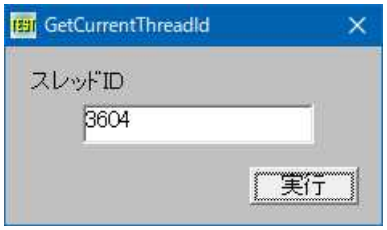
例では、自フォームの前後のZオーダーを持つウィンドウハンドルを取得します。
GetWindow ウィンドウハンドルを取得
GetWindowText ウィンドウのタイトル文字列を取得


```
'=====================================
'= 自フォーム前後のウィンドウハンドルを取得
'=    (GetNextWindow.bas)
'=====================================
#include "Windows.bi"

' 指定されたウィンドウと指定された関係にあるウィンドウのハンドルを取得
Declare Function Api_GetWindow& Lib "user32" Alias "GetWindow" (ByVal hWnd&, ByVal wCmd&)

' ウィンドウのタイトル文字列を取得
Declare Function Api_GetWindowText& Lib "user32" Alias "GetWindowTextA" (ByVal hWnd&,
ByVal lpString$, ByVal cch&)

' 方向を示す定数の宣言
#define GW_CHILD 5                          '基準となるウィンドウの子ウィンドウのうちトップレベルのウィンドウを検索

#define GW_HWNDFIRST 0                      '最前面のウィンドウを検索
#define GW_HWNDLAST 1                       '最背面のウィンドウを検索
#define GW_HWNDNEXT 2                       '基準となるウィンドウの次のウィンドウを検索
#define GW_HWNDPREV 3                       '基準となるウィンドウの前のウィンドウを検索
#define GW_OWNER 4                          '基準となるウィンドウのオーナーウィンドウを検索

Var Shared hBase As Long

Var Shared Text(3) As Object
Var Shared Button(1) As Object

For i = 0 To 3
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1)))
    Text(i).SetFontSize 14
    If i < 2 Then
        Button(i).Attach GetDlgItem("Button" & Trim$(Str$(i + 1)))
        Button(i).SetFontSize 14
    End If
Next

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    hBase = GethWnd

    Text(2).SetWindowText "&&H" & Hex$(whPrev)
    Text(3).SetWindowText GetWindowText
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var whPrev As Long
    Var Buffer As String * 516
```

```
    Var Ret As Long

    '手前のウィンドウのハンドルを取得
    whPrev = Api_GetWindow(hBase, GW_HWNDPREV)

    'ウィンドウハンドルを取得できたときは
    If whPrev <> 0 Then

        'ウィンドウハンドルを表示
        Text(2).SetWindowText "&&H" & Hex$(whPrev)

        'タイトルバーテキストをバッファへ取得
        Ret = Api_GetWindowText(whPrev, Buffer, Len(Buffer))

        'タイトルバーテキストを表示
        Text(3).SetWindowText Buffer

        '取得したハンドルを基準のハンドルに設定
        hBase = whPrev
    End If
End Sub

'=======================================
'=
'=======================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var whNext As Long
    Var Buffer As String * 516
    Var Ret As Long

    '背面のウィンドウのハンドルを取得
    whNext = Api_GetWindow(hBase, GW_HWNDNEXT)

    'ウィンドウハンドルを取得できたときは
    If whNext <> 0 Then

        'ウィンドウハンドルを表示
        Text(2).SetWindowText "&&H" & Hex$(whPrev)

        'タイトルバーテキストをバッファへ取得
        Ret = Api_GetWindowText(whNext, Buffer, Len(Buffer))

        'タイトルバーテキストを表示
        Text(3).SetWindowText Buffer

        '取得したハンドルを基準のハンドルに設定
        hBase = whNext
    End If
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

自分自身のスレッドIDを取得

GetCurrentThreadId 呼び出し側スレッドのスレッド識別子を取得




```
'=====================================
'= 自分自身のスレッドIDを取得
'=    (GetCurrentThreadId.bas)
'=====================================
#include "Windows.bi"

' 呼び出し側スレッドのスレッド識別子を取得
Declare Function Api_GetCurrentThreadId& Lib "kernel32" Alias "GetCurrentThreadId" ()

Var Shared Text1 As Object
Var Shared Text2 As Object
Var SHared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Text2.Attach GetDlgItem("Text2") : Text2.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var ThreadId As Long

    'スレッドIDを取得
    ThreadId = Api_GetCurrentThreadId()

    'スレッドIDを表示
    Text2.SetWindowtext Hex$(ThreadId)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

自分自身の擬似スレッドハンドルを取得

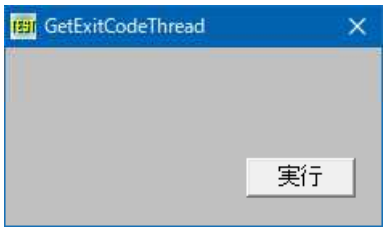
GetCurrentThread カレントスレッドの擬似ハンドルを取得


```
'=====================================
'= 自分自身の擬似スレッドハンドルを取得
'=    (GetCurrentThread.bas)
'=====================================
#include "Windows.bi"
```

```
' カレントスレッドの擬似ハンドルを取得
Declare Function Api_GetCurrentThread& Lib "kernel32" Alias "GetCurrentThread" ()

Var Shared Text1 As Object
Var Shared Text2 As Object
Var SHared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Text2.Attach GetDlgItem("Text2") : Text2.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Thread As Long

    '擬似スレッドハンドルを取得
    Thread = Api_GetCurrentThread()

    '擬似スレッドハンドルを表示
    Text2.SetWindowtext Hex$(Thread)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

自分自身のスレッドを終了させる

自分自身のスレッドを終了させます。
GetCurrentThread 自分自身の疑似スレッドハンドルを取得
ExitThread スレッドを終了
GetExitCodeThread 指定したスレッドの終了ステータスを取得

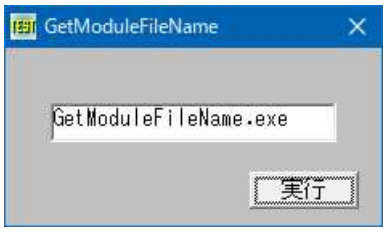



```
'=====================================
'= 自分自身のスレッドを終了させる
'=    (GetExitCodeThread.bas)
'=====================================
#include "Windows.bi"

' 自分自身の擬似スレッドハンドルを取得
Declare Function Api_GetCurrentThread& Lib "kernel32" Alias "GetCurrentThread" ()

' スレッドを終了
Declare Sub Api_ExitThread Lib "kernel32" Alias "ExitThread" (ByVal dwExitCode&)

' 指定したスレッドの終了ステータスを取得
Declare Function Api_GetExitCodeThread& Lib "kernel32" Alias "GetExitCodeThread" (ByVal
hThread&, lpExitCode&)
```

```
'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Api_ExitThread Api_GetExitCodeThread(Api_GetCurrentThread, 0)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

自分自身のファイル名を取得

GetModuleFileName ロードされている実行モジュールのフルパス名を取得
GetFileTitle パスからファイル名を取得

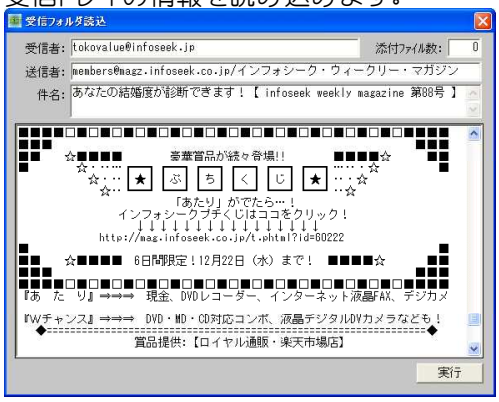



```
'=====================================
'= 自分自身のファイル名を取得
'=    (GetModuleFileName.bas)
'=====================================
#include "Windows.bi"

' ロードされている実行モジュールのフルパス名を取得
Declare Function Api_GetModuleFileName& Lib "Kernel32" Alias "GetModuleFileNameA" (ByVal
hModule&, ByVal lpFileName$, ByVal nSize&)

' パスからファイル名を取得
Declare Function Api_GetFileTitle& Lib "comdlg32" Alias "GetFileTitleA" (ByVal
lpszFile$, ByVal lpszTitle$, ByVal cbBuf%)

Var Shared Text1 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Buffer As String
    Var PathName As String
    Var FileName As String
    Var Ret As Long

    'パス名を取得
    Buffer = String$(260, Chr$(0))
    Ret = Api_GetModuleFileName(0, Buffer, Len(Buffer))
    PathName = Left$(Buffer, InStr(1, Buffer, Chr$(0)) - 1)
```

```
    'パス名からファイル名を抜き出す
    Buffer = String$(260, Chr$(0))
    Ret = Api_GetFileTitle(PathName, Buffer, Len(Buffer))
    FileName = Left$(Buffer, InStr(1, Buffer, Chr$(0)) - 1)

    Text1.SetWindowText FileName
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

受信トレイの読み込み

受信トレイの情報を読み込みます。



メールに関するコマンド	説明
MailFile関数	読み取り中のメールの添付ファイル名を返す
MailFileCount関数	読み取り中のメールの添付ファイル数を返す
MailFindNext関数	受信トレイ中に存在するメールのメールIDを取得する
MailItem関数	読み取り中のメール情報を返す
MailLogOff	メール機能からログオフする
MailLogOn	メール機能へログオンする
MailRead	受信トレイのメールを読み取る
MailRecip関数	読み取り中のメールの受信者情報を返す
MailRecipCount関数	読み取り中のメールの受信者数を返す
MailSend	メールを送信する
MailSender関数	読み取り中のメールの送信者情報を返す

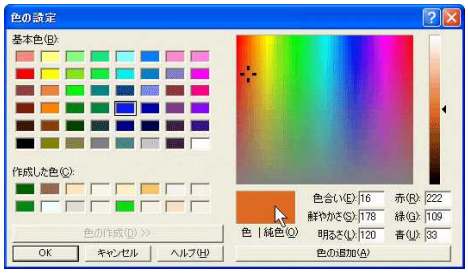
```
'=======================================
'= 受信トレイ読取(F-BASIC Sample 改変)
'=    (ReadMail.bas)
'=======================================
#include "windows.bi"
#include "file.bi"
#include "internet.bi"

' 指定された文字列と一致するクラス名とウィンドウ名を持つトップレベルウィンドウ( 親を持たないウィンドウ)のハンドルを返す。この関数は、子ウィンドウは探さない。検索では、大文字小文字は区別されない
Declare Function Api_FindWindow& Lib "user32" Alias "FindWindowA" (ByVal lpClassName$, ByVal lpWinDowName$)

' ウィンドウにメッセージを送信。この関数は、指定したウィンドウのウィンドウプロシージャが処理を終了するまで制御を返さない
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal wMsg&, ByVal wParam&, ByVal lParam&)

#define WM_CLOSE &H10                    'ウインドウ或いはアプリケーションをクローズされた
```

```
Var Shared Text(3) As Object
Var Shared Edit(4) As Object
Var Shared Button1 As Object

For i = 0 To 4
    If i < 4 Then
        Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1)))
        Text(i).SetFontSize 14
    End If
        Edit(i).Attach GetDlgItem("Edit" & Trim$(Str$(i + 1)))
        Edit(i).SetFontSize 14
Next i
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var Shared lpClassName$ As String
Var Shared lpCaption$ As String
Var Shared hWnd As Long

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Shell "Msimn.exe", , 2

    lpClassName$ = "Outlook Express Browser Class"
    lpCaption$ = "受信トレイ - Outlook Express"
    hWnd = Api_FindWindow(lpClassName$, lpCaption$)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()

    'メール機能へログイン
    MailLogOn , , 0
    SeedID$ = ""

    Do

        '受信フォルダに存在するメールのIDを取得
        MailID$ = MailFindNext(SeedID$, 0)
        If MailID$ = "" Then Exit Do

        'メールの読取
        MailRead MailID$

        'MailItem(0):件名 / MailItem(1):メッセージ / MailItem(2):受信日
        Letter$ = JConv$(MailItem(1), 0, 2)

        'MailRecip(0):受信者アドレス / MailRecip(1):受信者名 / MailRecip(2):受信者クラス
("TO","CC","BCC")
        Edit(0).SetWinDowText MailRecip(0)
        Edit(1).SetWinDowText MailSEnder(0) & "/" & MailSEnder(1)
        Edit(2).SetWinDowText MailItem(0)
        Edit(3).SetWinDowText Letter$

        'MailFileCount:添付ファイル数
        Edit(4).SetWinDowtext Format$(MailFileCount,"#####")
        SeedID$ = MailID$
    Loop

    MailLogOff
End Sub

'=====================================
'= 終了処理
'=====================================
```

```
Declare Sub MainForm_QueryClose edecl (Cancel%, ByVal Mode%)
Sub MainForm_QueryClose(Cancel%, ByVal Mode%)
    Var Ret As Long

    Cancel% = MessageBox(GetWinDowText, "終了しますか", 1, 1 )
    If Cancel% = 0 Then
        Ret = Api_SendMessage(hWnd, WM_CLOSE, 0, 0)
        End
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

上位・下位ワード(上位・下位バイト)取得

例では、マウスカーソルで指定した座標の色情報を取得し、上位ワード・下位ワードに分解、さらに上位バイト・下位バイトに分解しています。
GetCursorPos マウスカーソル(マウスポインタ)の現在の位置に相当するスクリーン座標を取得
LoByte 16ビット整数値の下位を取得
HiByte 16ビット整数値の上位を取得
LoWord 指定されたダブルワードの下位ワードを返す
HiWord 指定されたダブルワードの上位ワードを返す



図の色を選択した状態

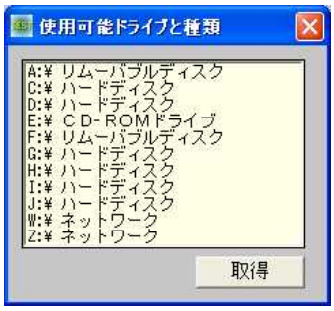
```
'=====================================
'= 上位・下位ワード(上位・下位バイト)取得
'=    (HiLoByte.bas)
'=====================================
#include "Windows.bi"

Type POINTAPI
    x   As Long
    y   As Long
End Type

' マウスカーソル(マウスポインタ)の現在の位置に相当するスクリーン座標を取得
Declare Function Api_GetCursorPos& Lib "user32" Alias "GetCursorPos" (lpPoint As
POINTAPI)

' 16ビット整数値の下位を取得
Declare Function Api_LoByte Lib "TLBINF32" Alias "lobyte" (ByVal Word%) As byte

' 16ビット整数値の上位を取得
Declare Function Api_HiByte Lib "TLBINF32" Alias "hibyte" (ByVal Word%) As byte

' 指定されたダブルワードの下位ワードを返す
Declare Function Api_LoWord% Lib "TLBINF32" Alias "loword" (ByVal DWord&)

' 指定されたダブルワードの上位ワードを返す
Declare Function Api_HiWord% Lib "TLBINF32" Alias "hiword" (ByVal DWord&)
```

```
' 指定された座標のピクセルのRGB値を取得
Declare Function Api_GetPixel& Lib "gdi32" Alias "GetPixel" (ByVal hDC&, ByVal X&, ByVal
Y&)

' ウィンドウ全体のデバイスコンテキストを取得。0(NULL)を指定すると、スクリーン全体のデバイスコンテキストを取
得
Declare Function Api_GetWindowDC& Lib "user32" Alias "GetWindowDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

Var Shared Timer1 As Object
Var Shared Text(14) As Object

Timer1.Attach GetDlgItem("Timer1")
For i = 0 To 14
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i+1)))
    Text(i).SetFontSize 14
Next

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Timer1.SetInterval 10
    Timer1.Enable -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var pAPI As POINTAPI
    Var Col As Long
    Var R As byte
    Var G As byte
    Var B As byte
    Var lDC As Long
    Var lw As Long
    Var hw As Long
    Var Ret As Long
    lDC = Api_GetWindowDC(0)

    Ret = Api_GetCursorPos(pAPI)

    Col = Api_GetPixel(lDC, pAPI.x, pAPI.y)
    lw = Api_LoWord(Col)
    hw = Api_HiWord(Col)

    R = Api_LoByte(lw)
    G = Api_HiByte(lw)
    B = Api_LoByte(hw)

    Text(6).SetWindowText "&&H" & Hex$(Col)
    Text(7).SetWindowText "&&H" & Hex$(lw)
    Text(8).SetWindowText "&&H" & Hex$(hw)

    Text(9).SetWindowText "&&H" & Hex$(R)
    Text(10).SetWindowText "&&H" & Hex$(G)
    Text(11).SetWindowText "&&H" & Hex$(B)

    Text(12).SetWindowText Str$(R)
    Text(13).SetWindowText Str$(G)
    Text(14).SetWindowText Str$(B)

    Ret = Api_ReleaseDC(0, lDC)
End Sub
```

```
'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

使用可能ドライブ一覧と種類の取得

使用可能ドライブ一覧と種類を取得します。
GetLogicalDrives 使用可能ドライブ一覧取得
GetDriveType 使用可能ドライブ種類取得




```
'=====================================
'=ドライブの種類を取得
'=    (DriveInfo.bas)
'=====================================
#include "Windows.bi"

' 有効なドライブの情報を取得。ディスクの挿入の有無やネットワークに接続されているかなどは関係なく、現在割り当てられている全てのドライブ一覧を返す
Declare Function Api_GetLogicalDrives& Lib "kernel32" Alias "GetLogicalDrives" ()

'ドライブのタイプを取得
Declare Function Api_GetDriveType& Lib "kernel32" Alias "GetDriveTypeA" (ByVal nDrive$)

#define DRIVE_CDROM 5                   'CD-ROMドライブ
#define DRIVE_FIXED 3                   '固定タイプ－主にハードディスク
#define DRIVE_NO_ROOT_DIR 1             'ルートディレクトリ無し
#define DRIVE_RAMDISK 6                 'RAMドライブ
#define DRIVE_REMOTE 4                  'ネットワーク
#define DRIVE_REMOVABLE 2               '取り外し可能タイプ－リムーバブルディスク
#define DRIVE_UNKNOWN 0                 'ドライブが不明

Var Shared List1 As Object
Var Shared Button1 As Object

List1.Attach GetDlgItem("List1") : List1.SetFontSize 12
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
sub Button1_on()
    Var i As integer                    'カウンタ
    Var DrvStr As String                'ドライブ名
    Var Drives As Long                  'ディスクドライブのビットマスク
    Var DrvType As Long                 'ドライブタイプ

    Drives = Api_GetLogicalDrives()     '現在利用可能なディスクドライブをビットマスク形式で取得
    If Drives = 0 Then Exit Sub         '関数の失敗
```

```
    List1.ResetContent

    For i = 0 To 25                          'A～Zドライブを検索する
        If (Drives And 1) = 1 Then
            DrvStr = Chr$(65 + i)            'ドライブ名(A～Z)に変換
            DrvStr = DrvStr & ":¥"
            DrvType = Api_GetDriveType(DrvStr)'そのドライブがどういうドライブタイプかをチェックする
            Select Case DrvType
                Case DRIVE_REMOVABLE
                    List1.AddString (DrvStr & " リムーバブルディスク")
                Case DRIVE_FIXED
                    List1.AddString (DrvStr & " ハードディスク")
                Case DRIVE_REMOTE
                    List1.AddString (DrvStr & " ネットワーク")
                Case DRIVE_CDROM
                    List1.AddString (DrvStr & " CD-ROMドライブ")
                Case DRIVE_RAMDISK
                    List1.AddString (DrvStr & " RAMドライブ")
                Case else
                    List1.AddString (DrvStr & " 不明ドライブ")
            End Select
        End If

        Drives = Drives ¥ 2                  '1ビットずつシフトさせていき、検索を続ける
    Next
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

条件付画像拡大縮小転送

StretchBltを使って画像を拡大(縮小)転送してみます。いろいろな条件を選択できるようですが識別しやすい条件のみ選択できるようにしてみました。
ビットマップファイルBitBlt.bmpとBitBlt2.bmpを用意します。例では120×80(ピクセル)
StretchBlt 画像拡大縮小転送
GetDC デバイスコンテキスト取得
ReleaseDC デバイスコンテキスト解放

フォーム設計(Picture1とPicture2を異なるサイズに設定します。)
テストに使用した画像(Picture1、Picture2)


```
'=======================================
'= 画像の拡大縮小転送
'=    (StretchBlt.bas)
'=======================================
#include "Windows.bi"
```

```
' 拡縮をともなうグラフィックデバイス間のイメージを転送
Declare Function Api_StretchBlt& Lib "gdi32" Alias "StretchBlt" (ByVal hDC&, ByVal X&,
ByVal Y&, ByVal nWidth&, ByVal nHeight&, ByVal hSrcDC&, ByVal xSrc&, ByVal ySrc&, ByVal
nSrcWidth&, ByVal nSrcHeight&, ByVal dwRop&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

#define SRCCOPY &HCC0020                    'コピー元をコピー
#define SRCPAINT &HEE0086                   'コピー元とコピー先をOR合成
#define SRCAND &H8800C6                     'コピー元とコピー先をAND合成
#define SRCINVERT &H660046                  'コピー元とコピー先をXOR合成
#define SRCERASE &H440328                   '転送先ビットマップを反転、その結果と転送元ビットマップを
                                            論理AND 演算子で結合
#define NOTSRCCOPY &H330008                 'コピー元を反転してコピー
#define NOTSRCERASE &H1100A6                'コピー元の色と、コピー先の色を論理OR演算子で結合し、さ
                                            らに反転
#define MERGECOPY &HC000CA                  'コピー元の色と、コピー先の色を論理AND演算子で結合
#define MERGEPAINT &HBB0226                 '反転した転送元ビットマップとパターンビットマップを論理
                                            OR演算子で結合
#define PATCOPY &HF00021                    '指定のパターンで描画先へコピー
#define PATINVERT &H5A0049                  'XORで指定のパターンと描画先の色を組み合わせる
#define PATPAINT &HF80A09                   '指定パターンの色と、コピー元の色を反転した色を論理OR
                                            演算子で結合し、結果をコピー先の色と論理OR演算子
#define DSTINVERT &H550009                  'コピー先を反転してコピー
#define BLACKNESS &H42                      'すべてを黒にしてコピー
#define WHITENESS &HFF0062                  'すべてを白にしてコピー
Var Shared Picture1 As Object
Var Shared Picture2 As Object
Var Shared Radio(4) As Object
Var Shared Bitmap As Object
BitmapObject Bitmap

Picture1.Attach GetDlgItem("Picture1")
Picture2.Attach GetDlgItem("Picture2")
For i = 0 To 4
    Radio(i).Attach GetDlgItem("Radio" & Trim$(Str$(i + 1)))
    Radio(i).SetFontSize 14
Next

Var Shared hDC1 As Long
Var Shared hDC2 As Long
Var Shared WK As Long

'=====================================
'=
'=====================================
Declare Sub PIC2_BMPSET edecl ()
Sub PIC2_BMPSET()
    Bitmap.LoadFile "TOKOs.bmp"
'   Picture2.StretchBitmap Bitmap, 0, 0, Picture2.GetWidth, Picture1.GetHeight
    Picture2.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    WK = &HCC0020
    Bitmap.LoadFile "TOM.bmp"
'   Picture1.StretchBitmap Bitmap, 0, 0, Picture1.GetWidth, Picture1.GetHeight
    Picture1.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject
    hDC1 = Api_GetDC(Picture1.GethWnd)
    hDC2 = Api_GetDC(Picture2.GethWnd)
```

```
End Sub

'=====================================
'= Picture1の画像をPicture2に条件付でコピーする
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    Ret = API_StretchBLT(hDC2, 0, 0, Picture2.GetWidth, Picture2.GetHeight, hDC1, 0, 0,
Picture1.GetWidth, Picture1.GetHeight, WK)
End Sub

'=====================================
'=
'=====================================
Declare Sub Radio1_on edecl ()
Sub Radio1_on()

    'Picture1をPicture2に転送
    Picture2.Cls
    WK = &HCC0020
End Sub

Declare Sub Radio2_on edecl ()
Sub Radio2_on()

    'Picture1とPicture2をOR合成
    Picture2.Cls
    WK = &HEE0086
    PIC2_BMPSET
End Sub

Declare Sub Radio3_on edecl ()
Sub Radio3_on()

    'Picture1とPicture2をAND合成
    Picture2.Cls
    WK = &H8800C6
    PIC2_BMPSET
End Sub

Declare Sub Radio4_on edecl ()
Sub Radio4_on()

    'Picture1とPicture2をXOR合成
    Picture2.Cls
    WK = &H660008
    PIC2_BMPSET
End Sub

Declare Sub Radio5_on edecl ()
Sub Radio5_on()

    'Picture1を反転して転送
    Picture2.Cls
    WK = &H330009
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_QueryClose edecl (Cancel%, ByVal Mode%)
Sub MainForm_QueryClose(Cancel%, ByVal Mode%)
    Var Ret As Long

    If Cancel% = 0 Then
        Ret = Api_ReleaseDC(Picture1.GethWnd, hDC1)
        Ret = Api_ReleaseDC(Picture2.GethWnd, hDC2)
        End
```

```
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

条件付画像転送(BitBlt)

BitBltを使って画像を転送してみます。いろいろな条件を選択できるようですが識別しやすい条件のみ選択できるようにしてみました。
ビットマップファイルBitBlt.bmpとBitBlt2.bmpを用意します。例では120×80(ピクセル)
BitBlt 画像転送
GetDC デバイス コンテキスト取得
ReleaseDC デバイス コンテキスト解放

テストに使用した画像(Picture1、Picture2)


```
'=====================================
'=  BitBlit(API)で画像転送
'=    (BitBlt_API.bas)
'=====================================
#include "Windows.bi"

' ビットブロック転送を行う。コピー元からコピー先のデバイスコンテキストへ、指定された長方形内の各ピクセルの色データをコピー
Declare Function Api_BitBlt& Lib "gdi32" Alias "BitBlt" (ByVal hDestDC&, ByVal X&, ByVal Y&, ByVal nWidth&, ByVal nHeight&, ByVal hSrcDC&, ByVal xSrc&, ByVal ySrc&, ByVal dwRop&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

#define SRCCOPY &HCC0020                'コピー元をコピー
#define SRCPAINT &HEE0086               'コピー元とコピー先をOR合成
#define SRCAND &H8800C6                 'コピー元とコピー先をAND合成
#define SRCINVERT &H660046              'コピー元とコピー先をXOR合成
#define SRCERASE &H440328               '転送先ビットマップを反転、その結果と転送元ビットマップを論理AND演算子で結合

#define NOTSRCCOPY &H330008             '色を反転して転送
#define NOTSRCERASE &H1100A6            'コピー元の色と、コピー先の色を論理OR演算子で結合し、さらに反転

#define MERGECOPY &HC000CA              'コピー元の色と、コピー先の色を論理AND演算子で結合
#define MERGEPAINT &HBB0226             '反転した転送元ビットマップとパターンビットマップを論理OR演算子で結合

#define PATCOPY &HF00021                '指定のパターンで描画先へコピー
```

```
#define PATINVERT &H5A0049                  'XORで指定のパターンと描画先の色を組み合わせる
#define PATPAINT &HF80A09                   '指定パターンの色と、コピー元の色を反転した色を論理OR
                                            演算子で結合し、結果をコピー先の色と論理OR演算子
#define DSTINVERT &H550009                  'コピー先を反転してコピー
#define BLACKNESS &H42                      'すべてを黒にしてコピー
#define WHITENESS &HFF0062                  'すべてを白にしてコピー

Var Shared Picture1 As Object
Var Shared Picture2 As Object
Var Shared Radio(4) As Object
Var Shared Group1 As Object
Var Shared Button1 As Object
Var Shared Bitmap As Object

BitmapObject Bitmap

Var Shared hDC1 As Long
Var Shared hDC2 As Long
Var Shared WK As Long

'=====================================
'=
'=====================================
Declare Sub Pic2_BmpSet edecl ()
Sub Pic2_BmpSet()
    Bitmap.LoadFile "tokovalue.bmp"
'   Picture2.StretchBitmap Bitmap, 0, 0, Picture2.GetWidth, Picture1.GetHeight
    Picture2.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    SetReDrawMode 1
    Picture1.Attach GetDlgItem("Picture1")
    Picture2.Attach GetDlgItem("Picture2")
    For i = 0 To 4
        Radio(i).Attach GetDlgItem("Radio" & Trim$(Str$(i + 1)))
        Radio(i).SetFontSize 14
    Next
    Group1.Attach GetDlgItem("Group1") : Group1.SetFontSize 14
    Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

    WK = &HCC0020
    Bitmap.LoadFile "fuji.bmp"
'   Picture1.StretchBitmap Bitmap, 0, 0, Picture1.GetWidth, Picture1.GetHeight
    Picture1.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject

    hDC1 = Api_GetDC(Picture1.GethWnd)
    hDC2 = Api_GetDC(Picture2.GethWnd)

    ShowWindow -1
    Cls
End Sub

'=====================================
'= Picture1の画像をPicture2に条件付でコピーする
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    Ret = Api_BitBlt(hDC2, 0, 0, Picture1.GetWidth, Picture1.GetHeight, hDC1, 0, 0, WK)
End Sub
```

```
'====================================
'=
'====================================
Declare Sub Radio1_on edecl ()
Sub Radio1_on()

    'Picture1をPicture2に転送
    Picture2.Cls
    WK = &HCC0020
End Sub

Declare Sub Radio2_on edecl ()
Sub Radio2_on()

    'Picture1とPicture2をOR合成
    Picture2.Cls
    WK = &HEE0086
    Pic2_BmpSet
End Sub

Declare Sub Radio3_on edecl ()
Sub Radio3_on()

    'Picture1とPicture2をAND合成
    Picture2.Cls
    WK = &H8800C6
    Pic2_BmpSet
End Sub

Declare Sub Radio4_on edecl ()
Sub Radio4_on()

    'Picture1とPicture2をXOR合成
    Picture2.Cls
    WK = &H660008
    Pic2_BmpSet
End Sub

Declare Sub Radio5_on edecl ()
Sub Radio5_on()

    'Picture1を反転して転送
    Picture2.Cls
    WK = &H330009
End Sub

'====================================
'=
'====================================
Declare Sub MainForm_QueryClose edecl (Cancel%, ByVal Mode%)
Sub MainForm_QueryClose(Cancel%, ByVal Mode%)
    Var Ret As Long

    If Cancel% = 0 Then
        Ret = Api_ReleaseDC(Picture1.GethWnd, hDC1)
        Ret = Api_ReleaseDC(Picture2.GethWnd, hDC2)
        End
    End If
End Sub

'====================================
'=
'====================================
While 1
    WaitEvent
Wend
Stop
End
```

使用できるCOMポートを検索

使用できるCOMポート(COM1～COM16)を調べます。
CreateFile ファイルの作成･オープン
CloseHandle オープンしているカーネルオブジェクトのハンドルをクローズ


```
'=======================================
'=  COMポートの存在を調べる
'=    (ComPort.bas)
'=======================================
#include "Windows.bi"

Type SECURITY_ATTRIBUTES
    nLength                 As Long
    lpSecurityDescriptor    As Long
    bInheritHandle          As Long
End Type

' ファイルの作成･オープン
Declare Function Api_CreateFile& Lib "kernel32" Alias "CreateFileA" (ByVal lFileName$,
ByVal dDesiredAccess&, ByVal dShareMode&, lSecurityAttributes As SECURITY_ATTRIBUTES,
ByVal dCreationDisposition&, ByVal dFlagsAndAttributes&, ByVal hTemplateFile&)

' オープンしているカーネルオブジェクトのハンドルをクローズ
Declare Function Api_CloseHandle& Lib "kernel32" Alias "CloseHandle" (ByVal hObject&)

#define FILE_SHARE_READ &H                  '他のアプリケーションからの読み込み可能
#define FILE_SHARE_WRITE &H2                '他のアプリケーションから書き込み可能
#define OPEN_EXISTING 3                     'ファイルをオープンする
#define FILE_ATTRIBUTE_NORMAL &H80          '他のファイル属性を持たない
Var Shared List1 As Object
Var Shared Button1 As Object

List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=======================================
'=
'=======================================
Declare Function ComAvailable(ComNum As Integer) As Integer
Function ComAvailable(ComNum As Integer) As Integer
    Var hCom As Long
    Var sa As SECURITY_ATTRIBUTES
    Var Ret As Long

    'ComPortを開く
    hCom = Api_CreateFile("COM" & Trim$(Str$(ComNum)) & "", 0, FILE_SHARE_READ Or
FILE_SHARE_WRITE, sa, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0)

    If hCom = True Then
        ComAvailable = False                'COMの存在なし
    Else
        ComAvailable = True                 'COMの存在あり
        Ret = Api_CloseHandle(hCom)         'ComPortクローズ
    End If
End Function

'=======================================
'=
'=======================================
```

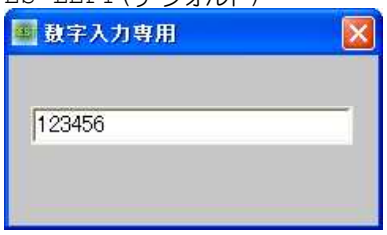
```
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var i As Integer

    For i = 1 To 16
        If ComAvailable(i) Then
            List1.AddString "COM" & Trim$(Str$(i)) & " OK"
        Else
            List1.AddString "COM" & Trim$(Str$(i)) & " NG"
        End If
    Next
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

数字入力専用

数字のみ入力できます。EditBoxプロパティで**数字のみ**を「あり」に設定した場合と同じですが‥
SetWindowLong 指定されたウィンドウの属性を変更
GetWindowLong 指定されたウィンドウに関しての情報を取得

例では、EditBoxプロパティで**数字のみ**を「なし」に設定し、APIで制御しています。

ES_LEFT(デフォルト)



ES_CENTER

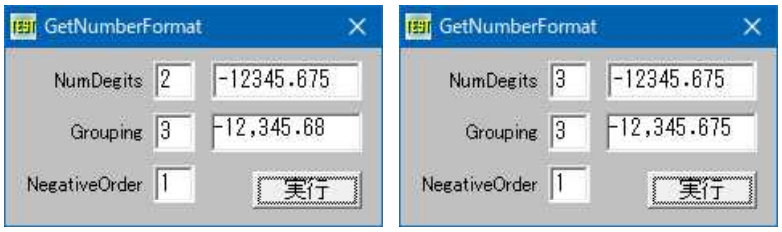
ES_RIGHT


```
'=====================================
'= 数字入力専用
'=    (NumInput.bas)
'=====================================
#include "Windows.bi"

' 指定されたウィンドウの属性を変更。また、拡張ウィンドウメモリの指定されたオフセットの32ビット値を書き換えることができる
Declare Function Api_SetWindowLong& Lib "user32" Alias "SetWindowLongA" (ByVal hWnd&,
ByVal nIndex&, ByVal dwNewLong&)

' 指定されたウィンドウに関しての情報を取得。また、拡張ウィンドウメモリから、指定されたオフセットにある32ビット値を取得することもできる
Declare Function Api_GetWindowLong& Lib "user32" Alias "GetWindowLongA" (ByVal hWnd&,
ByVal nIndex&)

#define GWL_STYLE -16                       'アプリケーションのインスタンスハンドル
#define ES_LEFT 0                           'テキストを左揃えする(デフォルト)
#define ES_CENTER 1                         'テキストを水平方向で中央に表示する
#define ES_RIGHT 2                          'テキストを右揃えする
#define ES_NUMBER &H2000                    '数値入力専用にする

Var Shared Edit1 As Object
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14

'=====================================
'=
'=====================================
```

```
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var Ret As Long

    Ret = Api_GetWindowLong(Edit1.GethWnd, GWL_STYLE)
    Ret = Api_SetWindowLong(Edit1.GethWnd, GWL_STYLE, Ret Or ES_NUMBER Or ES_RIGHT)

    Edit1.SetFocus
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

数字文字列の形式を書式化

GetNumberFormat 指定した数字文字列を、指定したロケール用にカスタマイズされた数字文字列として書式化

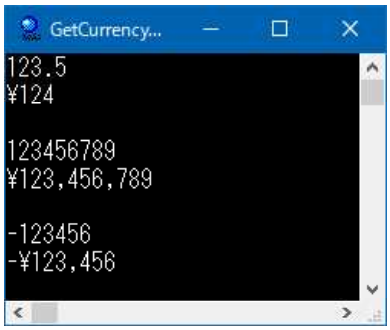



```
'=====================================
'= 数字文字列の形式を書式化
'=    (GetNumberFormat.bas)
'=====================================
#include "Windows.bi"

Type NUMBERFMT
    NumDigits         As Long              '小数点以下の数字の数
    LeadingZero       As Long              '小数点以下の数字の数がNumDigitsに満たない場合「0」
                                           を加える
    Grouping          As Long              'グループ(日本では千単位)の区切り文字数
    lpDecimalSep      As Long              '小数点を表す文字
    lpThousandSep     As Long              'グループの区切り文字
    NegativeOrder     As Long              '負数の表示方法
End Type

' 指定されたロケール用にカスタマイズされた数字文字列として数字文字列の形式を設定
Declare Function Api_GetNumberFormat& Lib "kernel32" Alias "GetNumberFormatA" (ByVal
Locale&, ByVal dwFlags&, ByVal lpValue$, lpFormat As NUMBERFMT, ByVal lpNumberStr$, ByVal
cchNumber&)

#define LOCALE_SYSTEM_DEFAULT &H400          'システムのデフォルトロケール
#define LOCALE_USER_DEFAULT &H800            '現在のユーザのデフォルトロケール

Var Shared Edit(3) As Object
Var Shared Text(3) As Object
Var Shared Button1 As Object

For i = 0 To 3
    Edit(i).Attach GetDlgItem("Edit" & Trim$(Str$(i + 1)))
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1)))
    Edit(i).SetFontSize 14
    If i < 1 Then
        Text(i).SetFontSize 14
    Else
```

```
        Text(i).SetFontSize 12
    End If
Next
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var num As String
    Var Buff As String
    Var nf As NUMBERFMT
    Var Ret As Long

    num = Edit(0).GetWindowText

    Buff = String$(255, Chr$(0))

    nf.NumDigits = Val(Edit(1).GetWindowText)
    nf.LeadingZero = 0
    nf.Grouping = Val(Edit(2).GetWindowText)
    nf.lpDecimalSep = StrAdr("." & Chr$(0))
    nf.lpThousandSep = StrAdr("," & Chr$(0))

    '-1.1 の場合 0:(1.1) 1:-1.1 2:- 1.1 3:1.1-  4:1.1 -
    nf.NegativeOrder = Val(Edit(3).GetWindowText)

    Ret = Api_GetNumberFormat(LOCALE_USER_DEFAULT, 0, num, nf, Buff, Len(Buff))
    Text(0).SetWindowText Left$(Buff, Ret)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

数値を通貨フォーマットで表示

入力した数値を指定された地域通貨フォーマットで表示します。
GetCurrencyFormat 通貨フォーマットを取得



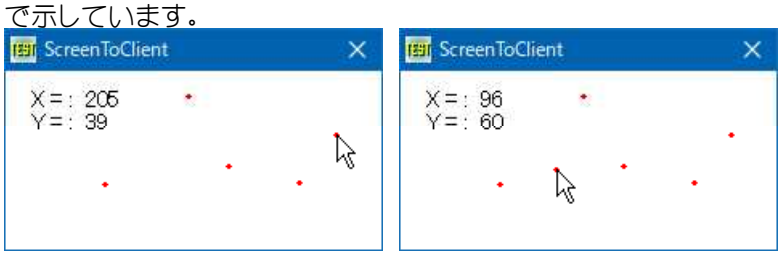

```
'=====================================
'= 数値を通貨フォーマットで表示
'=    (GetCurrencyFormat.bas)
'=====================================
#include "Windows.bi"
```

```
' 通貨フォーマットを取得
Declare Function Api_GetCurrencyFormat& Lib "kernel32" Alias "GetCurrencyFormatA" (ByVal
Locale&, ByVal dwFlags&, ByVal lpValue$, lpFormat As Any, ByVal lpCurrencyStr$, ByVal
cchCurrency&)

#define LOCALE_USER_DEFAULT &H800          '現在のユーザのデフォルトロケール

Var InputNum As String                     '入力文字列(数値)
Var Buffer As String                       'フォーマット
Var Ret As Long

InputNum = "1234567.89"
Buffer = String$(256, Chr$(0))
Ret = Api_GetCurrencyFormat(LOCALE_USER_DEFAULT, 0, InputNum, ByVal 0, Buffer,
len(Buffer))
Buffer = Left$(Buffer, InStr(Buffer, Chr$(0)) -1)

Print "入力文字列(数値)    : " & InputNum
Print "ロケールフォーマット: " & Buffer

Stop
End
```

スクリーン座標からクライアント座標に変換

マウスカーソルの現在の位置に相当するスクリーン座標を取得し、クライアント座標に変換します。
GetCursorPos スクリーン座標を取得
ScreenToClient 点座標をスクリーン座標からクライアント座標に変換

フォームをクリックし、そのスクリーン座標をクライアント座標に変換してその位置を表示、さらにクリックした位置を赤点で示しています。



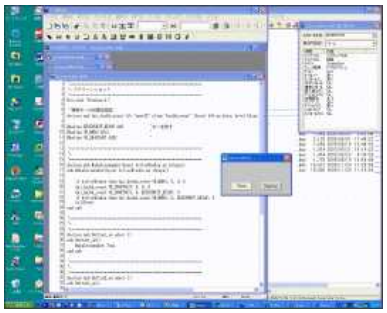

```
'=====================================
'= スクリーン座標からクライアント座標に変換
'=    (ScreenToClient.bas)
'=====================================
#include "Windows.bi"

Type POINTAPI
    X   As Long
    Y   As Long
End Type

' マウスカーソルの現在の位置に相当するスクリーン座標を取得
Declare Function Api_GetCursorPos& Lib "user32" Alias "GetCursorPos" (lpPoint As
POINTAPI)

' 点座標をスクリーン座標からクライアント座標に変換
Declare Function Api_ScreenToClient& Lib "user32" Alias "ScreenToClient" (ByVal hWnd&,
lpPoint As POINTAPI)

Var Shared Text1 As Object
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14

#define vbCrLf Chr$(13,10)

'=====================================
'=
'=====================================
```

```
Declare Sub Mainform_Click edecl ()
Sub Mainform_Click()
    Var pa As POINTAPI
    Var Ret As Long

    Ret = Api_GetCursorPos(pa)
    Ret = Api_ScreenToClient(GethWnd, pa)

    Text1.SetWindowText "X = : " & Str$(pa.X) & vbCrLf & "Y = : " & Str$(pa.Y)
    SetDrawWidth 3                              '目視し易いよう(老眼対策･･･)
    Pset(posi.X, posi.Y), 5
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

スクリーンショット(Ⅰ)

画面をキャプチャします。
keybd_event 特殊キーの状態を設定

「Form」をクリックすると左図のフォームが、「Desktop」をクリックすると右図のデスクトップ画面がキャプチャされます。



```
'=====================================
'= スクリーンショット
'=    (ScreenShot.bas)
'=====================================
#include "Windows.bi"

' 特殊キーの状態を設定
Declare Sub Api_keybd_event Lib "user32" Alias "keybd_event" (ByVal bVk As byte, ByVal
bScan As byte, ByVal dwFlags&, ByVal dwExtraInfo&)

#define KEYEVENTF_KEYUP &H2                 'キーを放す
#define VK_MENU &H12                        '[Menu]
#define VK_SNAPSHOT &H2C                    '[Snap Shot]

'=====================================
'=
'=====================================
Declare Sub MakeScreenshot(ByVal ActiveWindow As Integer)
Sub MakeScreenshot(ByVal ActiveWindow As Integer)

    If ActiveWindow Then Api_keybd_event VK_MENU, 0, 0, 0
    Api_keybd_event VK_SNAPSHOT, 0, 0, 0
    Api_keybd_event VK_SNAPSHOT, 0, KEYEVENTF_KEYUP, 0
    If ActiveWindow Then Api_keybd_event VK_MENU, 0, KEYEVENTF_KEYUP, 0
    CallEvent
End Sub
```

```
'=====================================
'= フォームをキャプチャ
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    MakeScreenshot True
End Sub

'=====================================
'= デスクトップをキャプチャ
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    MakeScreenshot False
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

スクリーンショット（II）

BitBlt ビットブロック転送を行う。コピー元からコピー先のデバイスコンテキストへ、指定された長方形内の各ピクセルの色データをコピー
StretchBlt 拡縮をともなうグラフィックデバイス間のイメージを転送
SetStretchBltMode 指定されたデバイスコンテキストのビットマップ伸縮モードを設定
GetDesktopWindow Windowsのデスクトップウィンドウを識別
GetDC 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
GetWindowDC ウィンドウ全体のデバイスコンテキストを取得
ReleaseDC デバイスコンテキストを解放

ディスプレイ解像度（2560x1440）


```
'=====================================
'= スクリーンショット（II）
'=    (BitBlt3.bas)
'=====================================
#include "Windows.bi"

' ビットブロック転送を行う。コピー元からコピー先のデバイスコンテキストへ、指定された長方形内の各ピクセルの色データをコピー
Declare Function Api_BitBlt& Lib "gdi32" Alias "BitBlt" (ByVal hDestDC&, ByVal X&, ByVal Y&, ByVal nWidth&, ByVal nHeight&, ByVal hSrcDC&, ByVal xSrc&, ByVal ySrc&, ByVal dwRop&)

' 拡縮をともなうグラフィックデバイス間のイメージを転送
Declare Function Api_StretchBlt& Lib "gdi32" Alias "StretchBlt" (ByVal hDC&, ByVal X&, ByVal Y&, ByVal nWidth&, ByVal nHeight&, ByVal hSrcDC&, ByVal xSrc&, ByVal ySrc&, ByVal nSrcWidth&, ByVal nSrcHeight&, ByVal dwRop&)
```

```
' 指定されたデバイスコンテキストのビットマップ伸縮モードを設定
Declare Function Api_SetStretchBltMode& Lib "gdi32" Alias "SetStretchBltMode" (ByVal
hDC&, ByVal nStretchMode&)

' Windowsのデスクトップウィンドウを識別。返されるポインタは、一時的なポインタ。後で使用するために保存しておくことはできない
Declare Function Api_GetDesktopWindow& Lib "user32" Alias "GetDesktopWindow" ()

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' ウィンドウ全体のデバイスコンテキストを取得
Declare Function Api_GetWindowDC& Lib "user32" Alias "GetWindowDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

#define SRCCOPY &HCC0020                    'そのまま転送
#define COLORONCOLOR 3                      '取り除く点の情報を保存することなく削除

Var Shared Picture1 As Object
Var Shared Check1 As Object
Var Shared Button1 As Object

Picture1.Attach GetDlgItem("Picture1")
Check1.Attach GetDlgItem("Check1") : Check1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub PrintScreen ()
Sub PrintScreen()
    Var phDC As Long
    Var hDesk As Long
    Var dhDC As Long
    Var Ret As Long

    phDC = Api_GetDC(Picture1.GethWnd)

    hDesk = Api_GetDesktopWindow()
    dhDC = Api_GetWindowDC(hDesk)

    If Check1.GetCheck = 1 Then
        Ret = Api_SetStretchBltMode(phDC, COLORONCOLOR)
        Ret = Api_StretchBlt(phDC, 0, 0, Picture1.GetWidth, Picture1.GetHeight, dhDC, 0,
0, GetDeviceCaps(8), GetDeviceCaps(10), SRCCOPY)
    Else
        Ret = Api_BitBlt(phDC, 0, 0, Picture1.GetWidth, Picture1.GetHeight, dhDC, 0, 0,
SRCCOPY)
    End If

    Ret = Api_ReleaseDC(Picture1.GethWnd, phDC)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var sx As Single
    Var sy As Single
    Var wpx As Single
    Var rsy As Single

    sx = GetDeviceCaps(8)
    sy = GetDeviceCaps(10)
    rsy = sy / sx
    wpx = Picture1.GetWidth
```

```
    Picture1.SetWindowSize wpx, wpx * rsy
    PrintScreen
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

スクリーンショットを平行四辺形で表示

スクリーンショットを平行四辺形で縮小表示します。
CreateDIBPatternBrushPt DIBとして定義されたパターンの論理ブラシを作成
PlgBlt 平行四辺形へのカラービット転送 (WindowsNT3.1以降、Windows98/Meはサポートしていません。)
PatBlt ビットパターンを作成 (WindowsNT3.1以降、Windows95以降)
SelectObject 指定されたデバイスコンテキストのオブジェクトを選択
DeleteObject システムリソースを解放
GetDC ディスプレイデバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放

例では、スクリーンショットを表示する左上の座標を指定し、平行四辺形の状態確認をしています。
Windows 10

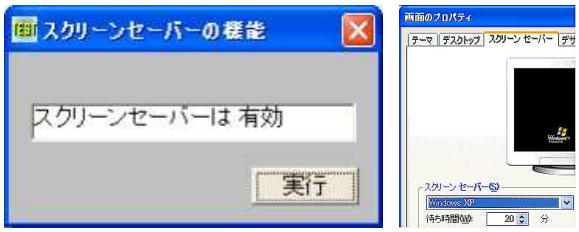

```
'=====================================
'=  スクリーンショットを平行四辺形で表示
'=    (Windows2000以降)
'=    (PlgBlt.bas)
'=====================================
#include "Windows.bi"

#define BI_RGB 0
#define DIB_RGB_COLORS 0                    'RGBカラーテーブル
#define DIB_PAL_COLORS 1                    'パレットカラーテーブル
#define PATCOPY &HF00021
#define PATINVERT &H5A0049
#define PATPAINT &HFB0A09

Type POINTAPI
    x   As Long
    y   As Long
End Type

Type BITMAPINFOHEADER                       '40バイト
    biSize            As Long
    biWidth           As Long
    biHeight          As Long
    biPlanes          As Integer
    biBitCount        As Integer
    biCompression     As Long
    biSizeImage       As Long
    biXPelsPerMeter   As Long
```

```
    biYPelsPerMeter   As Long
    biClrUsed         As Long
    biClrImportant    As Long
End Type

Type BITMAPINFO
    bmiHeader As BITMAPINFOHEADER
End Type

Type tBITMAP
    Header     As BITMAPINFO
    Bytes(63)  As byte
End Type

' DIBとして定義されたパターンの論理ブラシを作成
Declare Function Api_CreateDIBPatternBrushPt& Lib "gdi32" Alias
"CreateDIBPatternBrushPt" (lpPackedDIB As Any, ByVal iUsage&)

' 転送元DCの指定された四角形から、指定されたDCの指定した平行四辺形へ、カラーデータのビットブロック転送を
行う
Declare Function Api_PlgBlt& Lib "gdi32" Alias "PlgBlt" (ByVal hdcDest&, lpPoint As
POINTAPI, ByVal hdcSrc&, ByVal nXSrc&, ByVal nYSrc&, ByVal nWidth&, ByVal nHeight&, ByVal
hbmMask&, ByVal xMask&, ByVal yMask&)

' ビットパターンを作成
Declare Function Api_PatBlt& Lib "gdi32" Alias "PatBlt" (ByVal hDC&, ByVal x&, ByVal y&,
ByVal nWidth&, ByVal nHeight&, ByVal dwRop&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

' 指定されたデバイスコンテキストのオブジェクトを選択
Declare Function Api_SelectObject& Lib "gdi32" Alias "SelectObject" (ByVal hDC&, ByVal
hObject&)

' ペン、ブラシ、フォント、ビットマップ、リージョン、パレットのいずれかの論理オブジェクトを削除し、そのオブジェクトに
関連付けられていたすべてのシステムリソースを解放。オブジェクトを削除した後は、指定されたハンドルは無効になる
Declare Function Api_DeleteObject& Lib "gdi32" Alias "DeleteObject" (ByVal hObject&)

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    cls
    Var hBrush As Long
    Var tBr As tBITMAP
    Var hOld As Long
    Var Pt(2) As POINTAPI
    Var Ret As Long
    Var hDC As Long

    '平行四辺形の傾き指定
    Pt(0).x = Val(GetDlgItemText("Edit1"))'左上x(可変)
    Pt(0).y = Val(GetDlgItemText("Edit2"))' 〃 y(可変)
    Pt(1).x = 300                         '右上x
    Pt(1).y = 0                           ' 〃 y
    Pt(2).x = 0                           '左下x
    Pt(2).y = 300                         ' 〃 y

    hDC = Api_GetDC(GethWnd)              'フォームのデバイスコンテキスト

    'スクリーンショットのリサイズ
    Ret = Api_PlgBlt(hDC, Pt(0), Api_GetDC(0), 0, 0, GetDeviceCaps(8), GetDeviceCaps(10),
ByVal 0&, ByVal 0&, ByVal 0&)
```

```
    'tBITMAP構造体の初期化
    tBr.Header.bmiHeader.biSize = Len(tBr.Header.bmiHeader)
    tBr.Header.bmiHeader.biCompression = BI_RGB
    tBr.Header.bmiHeader.biHeight = 8
    tBr.Header.bmiHeader.biPlanes = 1
    tBr.Header.bmiHeader.biWidth = 8
    tBr.Header.bmiHeader.biBitCount = 1

    For i = 0 To 7
        tBr.Bytes(i) = 128
    Next i

    hBrush = Api_CreateDIBPatternBrushPt(tBr, DIB_RGB_COLORS)'パターンブラシ作成
    hOld = Api_SelectObject(hDC, hBrush)            'フォームDCに対するBRUSH選択
    Ret = Api_PatBlt(hDC, 0, 0, 30, 30, PATCOPY)    'パターンブロック転送
    Ret = Api_DeleteObject(Api_SelectObject(hDC, hOld))  'パターンブラシを元に戻す
    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

スクリーンセーバー機能の有効無効を判定

SystemParametersInfo システム全体に関するパラメータを取得・設定
SPI_GETSCREENSAVEACTIVE(16) スクリーンセーバー機能が有効かどうか調べる
SPI_SETSCREENSAVEACTIVE(17) スクリーンセーバーを有効・無効に




```
'=======================================
'= スクリーンセーバー機能の有効無効を判定
'=    (SPI_GETSCREENSAVEACTIVE.bas)
'=======================================
#include "Windows.bi"

#define SPI_GETSCREENSAVEACTIVE 16          'スクリーンセーバー機能が有効かどうか調べる
#define SPI_SETSCREENSAVEACTIVE 17          'スクリーンセーバーを有効・無効に

' システム全体に関するパラメータを取得・設定
Declare Function Api_SystemParametersInfo& Lib "user32" Alias "SystemParametersInfoA"
(ByVal uiAction&, ByVal uiParam&, pvParam As Any, ByVal fWinIni&)

Var Shared Text1 As Object
Var Shared Button1 As Object
```

```
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var ScrnSaverActive  As Long
    Var Ret As Long

    'スクリーンセーバー機能の有効無効を取得
    Ret = Api_SystemParametersInfo(SPI_GETSCREENSAVEACTIVE, 0, ScrnSaverActive, 0)

    'スクリーンセーバー機能の有効無効を表示
    If ScrnSaverActive Then
        Text1.SetWindowText "スクリーンセーバーは 有効"
    Else
        Text1.SetWindowText "スクリーンセーバーは 無効"
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

スクリーンセーバーの確認

設定されているスクリーンセーバーを確認します。
SendMessage ウィンドウメッセージを送信
WM_SYSCOMMAND(&H112) システムメニューが操作された
SC_SCREENSAVE(&HF140) スクリーンセーバーを実行するメッセージ

「確認」ボタンをクリックすると、設定されている場合はスクリーンセーバーが表示されます。設定されていない場合は画面の変化はありません。

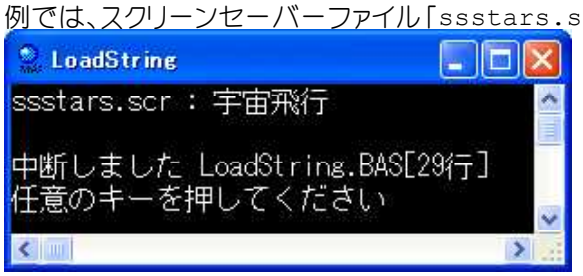
```
'=====================================
'= スクリーンセーバーの確認
'=    (ScreenSaver.bas)
'=====================================
#include "Windows.bi"

' ウィンドウにメッセージを送信。この関数は、指定したウィンドウのウィンドウプロシージャが処理を終了するまで制御を返さない
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal wMsg&, ByVal wParam&, ByVal lParam&)

#define WM_SYSCOMMAND &H112                 'システムメニューが操作された
#define SC_SCREENSAVE &HF140                'スクリーンセーバーを実行するメッセージ
```

```
'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    Ret = Api_SendMessage(GethWnd, WM_SYSCOMMAND, SC_SCREENSAVE, 0)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

スクリーンセーバーファイルから文字列を抽出

LoadLibrary DLLをロード
FreeLibrary ロードしたDLLの解放
LoadString 実行ファイルに埋めこまれた文字列リソースをロード

例では、スクリーンセーバーファイル「ssstars.scr」に埋め込まれている文字列「宇宙飛行」を抽出しています。




```
'=====================================
'= スクリーンセーバーファイルから文字列を抽出
'=    (LoadString.bas)
'=====================================

' DLLをロード
Declare Function Api_LoadLibrary& Lib "kernel32" Alias "LoadLibraryA" (ByVal
lpLibFileName$)

' ロードしたDLLの解放
Declare Sub Api_FreeLibrary Lib "kernel32" Alias "FreeLibrary" (ByVal hLibModule&)

' 実行ファイルに埋めこまれた文字列リソースをロード
Declare Function Api_LoadString& Lib "user32" Alias "LoadStringA" (ByVal hInstance&,
ByVal wID&, ByVal lpBuffer$, ByVal nBufferMax&)

Var FileName As String
Var Instance As Long
Var Buffer As String * 255
Var Ret As Long

FileName = "ssstars.scr"

Instance = Api_LoadLibrary("c:¥windows¥system32¥" & FileName)

Ret = Api_LoadString(Instance, 1, Buffer, 255)

Print FileName & " : " & Left$(Buffer, InStr(Buffer, Chr$(0)) - 1)

Api_FreeLibrary(Instance)
```

```
Stop
End
```

スクリーンセーバー実行までの待ち時間を取得・設定

SystemParametersInfo システム全体に関するパラメータを取得・設定
SPI_GETSCREENSAVETIMEOUT(14) スクリーンセーバー実行までの待ち時間を取得
SPI_SETSCREENSAVETIMEOUT(15) スクリーンセーバー実行までの待ち時間を設定
SPIF_SENDWININICHANGE(&H2) 全てのアプリケーションに通知して更新
SPIF_UPDATEINIFILE(&H1) ユーザープロファイルの更新を指定

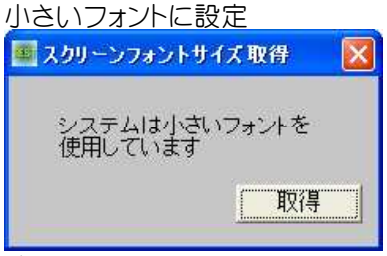

```
'=====================================
'= スクリーンセーバー実行までの待ち時間を取得・設定
'=    (SPI_GETSCREENSAVETIMEOUT.bas)
'=====================================
#include "Windows.bi"

' システム全体に関するパラメータを取得・設定
Declare Function Api_SystemParametersInfo& Lib "user32" Alias "SystemParametersInfoA"
(ByVal uiAction&, ByVal uiParam&, pvParam As Any, ByVal fWinIni&)

#define SPI_GETSCREENSAVETIMEOUT 14          'スクリーンセーバー実行までの待ち時間を取得
#define SPI_SETSCREENSAVETIMEOUT 15          'スクリーンセーバー実行までの待ち時間を設定
#define SPIF_SENDWININICHANGE &H2            '全てのアプリケーションに通知して更新
#define SPIF_UPDATEINIFILE &H1               'ユーザープロファイルの更新を指定
Var Shared Text1 As Object
Var Shared Edit1 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Timeout As Long
    Var Ret As Long

    'スクリーンセーバー機能の実行までの待ち時間を取得
    Ret = Api_SystemParametersInfo(SPI_GETSCREENSAVETIMEOUT, 0, Timeout, 0)

    '待ち時間を表示
    Edit1.SetWindowText Str$(Timeout)
End Sub

'=====================================
'=
'=====================================
```

```
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Timeout As Long
    Var Ret As Long

    TimeOut = Val(Edit1.GetWindowText)

    'スクリーンセーバー機能の実行までの待ち時間を設定
    Ret = Api_SystemParametersInfo(SPI_SETSCREENSAVETIMEOUT, Timeout, ByVal CLng(0),
SPIF_UPDATEINIFILE Or SPIF_SENDWININICHANGE)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

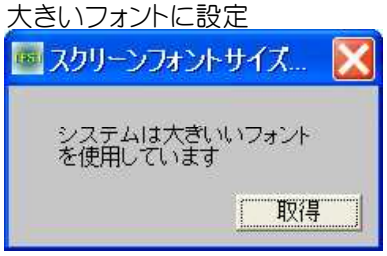
スクリーンフォントサイズを取得

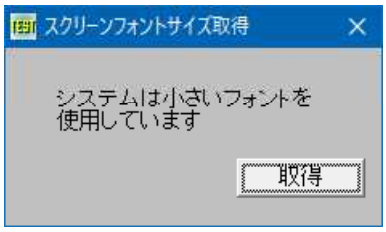
スクリーンフォントサイズを取得します。
GetDesktopWindow デスクトップハンドルを取得
GetDeviceCaps デバイス固有の情報を取得

小さいフォントに設定

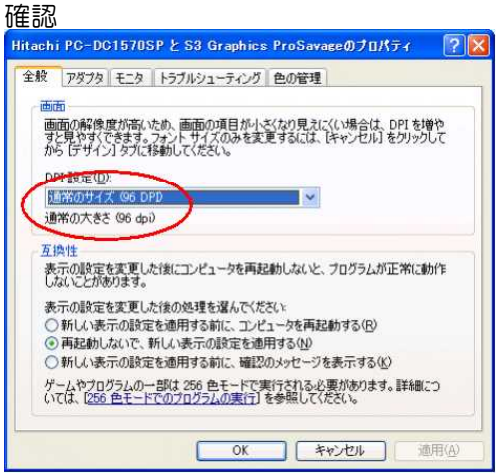


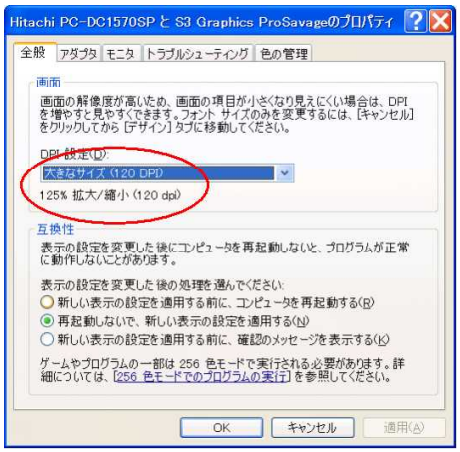
大きいフォントに設定





確認






```
'=====================================
'= スクリーンフォントサイズを取得
'=    (ScreenFontSize.bas)
'=====================================
#include "Windows.bi"

#define LOGPIXELSX 88
#define LOGPIXELSY 90

' Windowsのデスクトップウィンドウを識別。返されるポインタは、一時的なポインタ。後で使用するために保存しておくことはできない
Declare Function Api_GetDesktopWindow& Lib "user32" Alias "GetDesktopWindow" ()
```

```
' デバイス固有の情報を取得
Declare Function Api_GetDeviceCaps& Lib "gdi32" Alias "GetDeviceCaps" (ByVal hDC&, ByVal
nIndex&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

Var Shared Text1 As Object
Var Shared button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Function IsScreenFontSmall() As Integer
Function IsScreenFontSmall() As Integer
    Var hWndDesk As Long
    Var hDCDesk As Long
    Var logPix As Long
    Var Ret As Long

    'デスクトップのハンドル取得
    hWndDesk = Api_GetDesktopWindow()

    'デスクトップのデバイスコンテキスト取得
    hDCDesk = Api_GetDC(hWndDesk)

    '水平の論理ピクセル取得
    logPix = Api_GetDeviceCaps(hDCDesk, LOGPIXELSX)

    'デバイスコンテキスト解放
    Ret = Api_ReleaseDC(hWndDesk, hDCDesk)

    'logPixの値が96である場合システムは小さいフォントを使用している
    IsScreenFontSmall = logPix
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    If IsScreenFontSmall = 96 Then
        Text1.SetWindowText "システムは小さいフォントを使用しています"
    Else
        Text1.SetWindowText "システムは大きいいフォントを使用しています"
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

スクロールバーのスライダ位置等を取得

スクロールバーのスライダ位置および範囲を取得します。
GetScrollRange スクロール範囲を取得

GetScrollPos スクロールバーのスライダ位置を取得

例では、EditBoxの垂直スクロールバーのスライダ位置と範囲を取得しています。

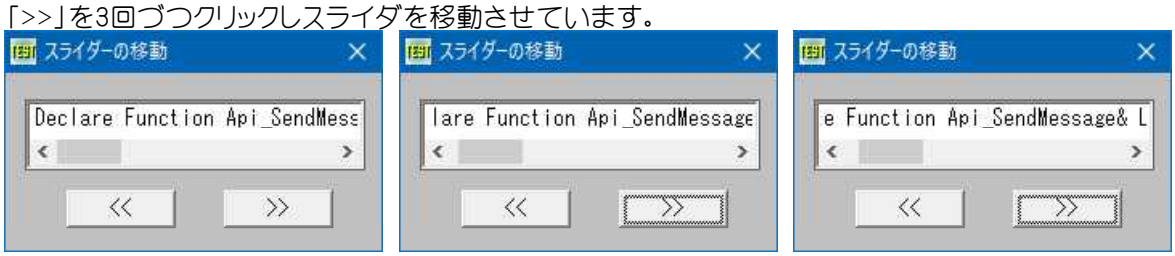

```
'=====================================
'= スクロールバーのスライダ位置等を取得
'=    (GetScrollPos.bas)
'=====================================
#include "Windows.bi"

' スクロール範囲を取得
Declare Function Api_GetScrollRange& Lib "user32" Alias "GetScrollRange" (ByVal hWnd&,
ByVal nBar&, lpMinPos&, lpMaxPos&)

' スクロールバーのスライダ位置を取得
Declare Function Api_GetScrollPos& Lib "user32" Alias "GetScrollPos" (ByVal hWnd&, ByVal
nBar&)

#define SB_HORZ 0                           '標準スクロールバーの水平
#define SB_VERT 1                           '標準スクロールバーの垂直
#define SB_CTL 2                            'スクロールバーコントロールの情報を設定
#define vbCrLf (Chr$(13) & Chr$(10))        'キャリッジリターンとラインフィード(¥r¥n)
Var Shared Edit1 As Object
Var Shared Text1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Function GetScrollbarPos(ByVal hWnd As Long, ByVal Flag As Long) As Long
Function GetScrollbarPos(ByVal hWnd As Long, ByVal Flag As Long) As Long
    Var Ret As Long

    Ret = Api_GetScrollPos(Edit1.GethWnd, Flag)

    If Ret <> 0 Then
        GetScrollbarPos = Ret
    Else
        GetScrollbarPos = 0
    End If
End Function

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var txt As String
    Var i As Long

    For i = 1 To 100
        If i < 100 Then
            txt = txt & Right$(Str$(1000 + i), 3) & vbCrLf
        Else
            txt = txt & Right$(Str$(1000 + i), 3)
        End If
    Next
    Edit1.SetWindowText txt
End Sub
```

```
'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var sMax As Long
    Var sMin As Long
    Var nPos As Long
    Var Ret As Long

    nPos = GetScrollbarPos(Edit1.GethWnd, SB_VERT)

    Ret = Api_GetScrollRange(Edit1.GethWnd, SB_VERT, sMin, sMax)
    Text1.SetWindowText Str$(sMin) & "-" & Str$(nPos) & "-" & Str$(sMax)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

スクロールバーのスライダを移動させる

スクロールバーのスライダを移動設定します。
SendMessage ウィンドウにメッセージを送信
WM_HSCROLL(&H114) 水平スクロールバーを調整
SB_LINELEFT(0) 左矢印がクリックされた
SB_LINERIGHT(1) 右矢印がクリックされた

「>>」を3回づつクリックしスライダを移動させています。



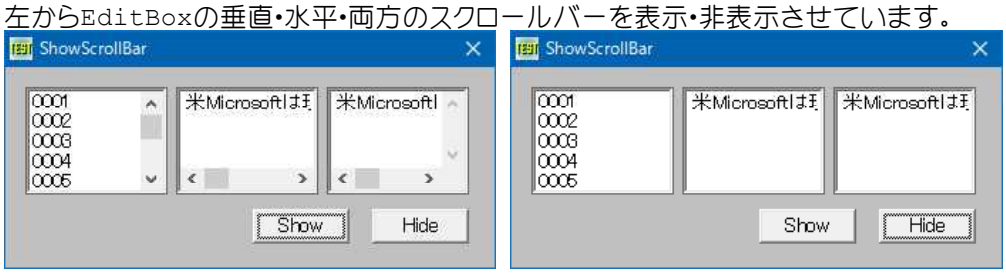

```
'=====================================
'= スクロールバーのスライダを移動させる
'=    (ScrollMove.bas)
'=====================================
#include "Windows.bi"

' ウィンドウにメッセージを送信。この関数は、指定したウィンドウのウィンドウプロシージャが処理を終了するまで制御を返さない
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal wMsg&, ByVal wParam&, ByVal lParam&)

#define WM_HSCROLL &H114                    '水平スクロールバーを調整している
#define SB_LINELEFT 0                       '左矢印がクリックされた
#define SB_LINERIGHT 1                      '右矢印がクリックされた

Var Shared Edit1 As Object
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Function hScrollLeft()
Function hScrollLeft()
    Var Ret As Long
```

```
    Ret = Api_SendMessage(Edit1.GethWnd, WM_HSCROLL, SB_LINELEFT, ByVal 0)
End Function

'=====================================
'=
'=====================================
Declare Function hScrollRight()
Function hScrollRight()
    Var Ret As Long

    Ret = Api_SendMessage(Edit1.GethWnd, WM_HSCROLL, SB_LINERIGHT, ByVal 0)
End Function

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Edit1.SetWindowText "Declare Function Api_SendMessage& Lib ""user32"" Alias
""SendMessageA"" (ByVal hWnd&, ByVal wMsg&, ByVal wParam&, ByVal lParam&)"
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    Ret = hScrollLeft
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Ret As Long

    Ret = hScrollRight
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

スクロールバーの表示･非表示

スクロールバーを表示･非表示します。
ShowScrollBar スクロールバーを表示

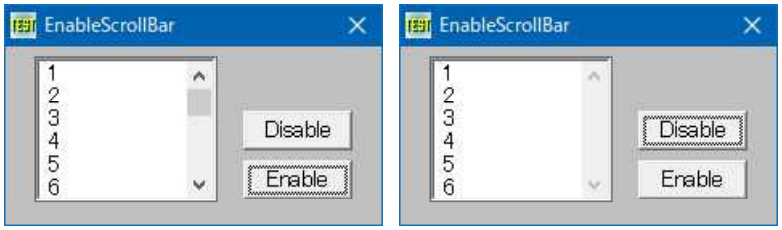
左からEditBoxの垂直･水平･両方のスクロールバーを表示･非表示させています。



```
'====================================
'= スクロールバーの表示･非表示
'=    (ShowScrollBar.bas)
'====================================
#include "Windows.bi"

' スクロールバーを表示
Declare Function Api_ShowScrollBar& Lib "user32" Alias "ShowScrollBar" (ByVal hWnd&,
ByVal wBar&, ByVal bShow&)

#define SB_HORZ 0                              '標準スクロールバーの水平
#define SB_VERT 1                              '標準スクロールバーの垂直
#define SB_BOTH 3                              '標準スクロールバーの水平･垂直両方

Var Shared Edit(1) As Object
Var Shared Button(1) As Object
Var Shared List1 As Object

For i = 0 To 1
    Edit(i).Attach GetDlgItem("Edit" & Trim$(Str$(i + 1)))
    Button(i).Attach GetDlgItem("Button" & Trim$(Str$(i + 1)))
    Edit(i).SetFontSize 14
    Button(i).SetFontSize 14
Next
List1.Attach GetDlgItem("List1") : List1.SetFontSize 14

'====================================
'=
'====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    For i = 1 To 100
        List1.AddString Right$(Str$(10000 + i), 4)
    Next
End Sub

'====================================
'= スクロールバー表示
'====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    Ret = Api_ShowScrollBar(List1.GethWnd, SB_VERT, True)
    Ret = Api_ShowScrollBar(Edit(0).GethWnd, SB_HORZ, True)
    Ret = Api_ShowScrollBar(Edit(1).GethWnd, SB_BOTH, True)
End Sub

'====================================
'= スクロールバー非表示
'====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Ret As Long

    Ret = Api_ShowScrollBar(List1.GethWnd, SB_VERT, False)
    Ret = Api_ShowScrollBar(Edit(0).GethWnd, SB_HORZ, False)
    Ret = Api_ShowScrollBar(Edit(1).GethWnd, SB_BOTH, False)
End Sub

'====================================
'=
'====================================
While 1
    WaitEvent
Wend
Stop
End
```

スクロールバーの有効･無効化

スクロールバーを有効化･無効化と切り替えます。
EnableScrollBar スクロールバーの有効･無効化

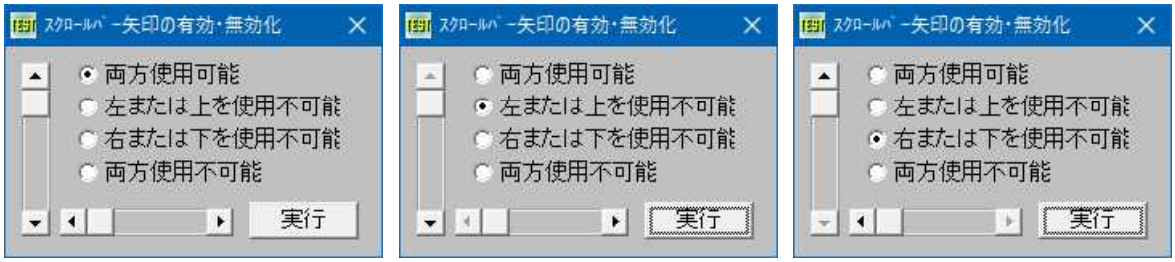


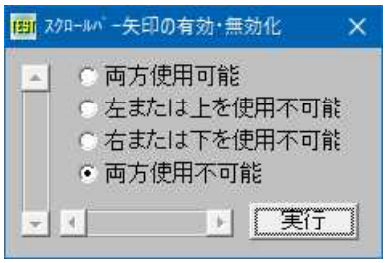

```
'=====================================
'= スクロールバーの有効･無効化
'=    (EnableScrollBar.bas)
'=====================================
#include "Windows.bi"

' スクロールバーの有効･無効化
Declare Function Api_EnableScrollBar& Lib "user32" Alias "EnableScrollBar" (ByVal hWnd&,
ByVal wSBflags&, ByVal wArrows&)

#define SB_BOTH 3                       '標準スクロールバーの水平･垂直両方
#define SB_CTL 2                        'スクロールバーコントロールの情報を設定
#define SB_HORZ 0                       '標準スクロールバーの水平
#define SB_VERT 1                       '標準スクロールバーの垂直

#define ESB_ENABLE_BOTH &H0             '両方向有効
#define ESB_DISABLE_LTUP &H1            '左･上方向無効
#define ESB_DISABLE_RTDN &H2            '右･下方向無効
#define ESB_DISABLE_BOTH &H3            '両方向無効

Var Shared List1 As Object
Var Shared Button1 As Object
Var Shared Button2 As Object

List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var i As Long

    For i = 1 To 20
        List1.AddString Str$(i)
    Next
End Sub

'=====================================
'= スクロールバーの無効化
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    Ret = Api_EnableScrollBar(List1.GethWnd, SB_VERT, ESB_DISABLE_BOTH)
End Sub

'=====================================
'= スクロールバーの有効化
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
```

```
    Var Ret As Long

    Ret = Api_EnableScrollBar(List1.GethWnd, SB_VERT, ESB_ENABLE_BOTH)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

スクロールバー矢印の有効･無効化

スクロールバーの矢印を有効化･無効化に設定します。
SendMessage ウィンドウにメッセージを送信
SBM_ENABLE_ARROWS(&HE4) スクロールバーの矢印を使用可能または使用不能にする
ESB_DISABLE_BOTH(&H3) 両方向無効
ESB_DISABLE_LTUP(&H1) 左･上方向無効
ESB_DISABLE_RTDN(&H2) 右･下方向無効
ESB_ENABLE_BOTH(&H0) 両方向有効






```
'=====================================
'= スクロールバー矢印の有効･無効化
'=    (ScrlEnableArrows.bas)
'=====================================
#include "Windows.bi"

' ウィンドウにメッセージを送信。この関数は、指定したウィンドウのウィンドウプロシージャが処理を終了するまで制御を返さない
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal wMsg&, ByVal wParam&, lParam As Any)

#define SBM_ENABLE_ARROWS &HE4              'スクロールバーの矢印を使用可能または使用不能にする
#define ESB_DISABLE_BOTH &H3                '両方向無効
#define ESB_DISABLE_LTUP &H1                '左･上方向無効
#define ESB_DISABLE_RTDN &H2                '右･下方向無効
#define ESB_ENABLE_BOTH &H0                 '両方向有効

Var Shared Radio(3) As Object
Var Shared HScroll1 As Object
Var Shared VScroll1 As Object
Var Shared Button1 As Object

For i = 0 To 3
```

```
    Radio(i).Attach GetDlgItem("Radio" & Trim$(Str$(i + 1)))
    Radio(i).SetFontSize 14
Next
HScroll1.Attach GetDlgItem("HScroll1")
VScroll1.Attach GetDlgItem("VScroll1")
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Function Arrows bdecl () As Integer
Function Arrows()
    Arrows = Val(Mid$(GetDlgRadioSelect("Radio1"), 6)) -1
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    'スクロールバーの使用状態を設定
    Ret = Api_SendMessage(HScroll1.GethWnd, SBM_ENABLE_ARROWS, Arrows, ByVal CLng(0))
    Ret = Api_SendMessage(VScroll1.GethWnd, SBM_ENABLE_ARROWS, Arrows, ByVal CLng(0))
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

スクロールバーを左右入れ替える

GetWindowLong 指定されたウィンドウに関しての情報を取得
SetWindowLong 指定されたウィンドウの属性を変更


```
'=====================================
'= スクロールバーを左右入れ替える
'=    (SetWindowLong5.bas)
'=====================================
#include "Windows.bi"

#define GWL_EXSTYLE -20                     '拡張ウィンドウスタイル
#define WS_EX_RIGHT &H1000                  '右揃えされたプロパティを持つウィンドウを作成
#define WS_EX_LEFTSCROLLBAR &H4000          '垂直スクロールバーがクライアント領域の左側に置かれる

' 指定されたウィンドウに関しての情報を取得
Declare Function Api_GetWindowLong& Lib "user32" Alias "GetWindowLongA" (ByVal hWnd&,
ByVal nIndex&)

' 指定されたウィンドウの属性を変更
Declare Function Api_SetWindowLong& Lib "user32" Alias "SetWindowLongA" (ByVal hWnd&,
ByVal nIndex&, ByVal dwNewLong&)
```

```
Var Shared List1 As Object
Var Shared Check1 As Object
Var Shared Check2 As Object

List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
Check1.Attach GetDlgItem("Check1") : Check1.SetFontSize 14
Check2.Attach GetDlgItem("Check2") : Check2.SetFontSize 14

Var Shared glnglpOriginalWndProc As Long
Var Shared glngOriginalhWnd As Long

'=====================================
'=
'=====================================
Declare Sub ListScrollAlign(ByVal Align As Long)
Sub ListScrollAlign(ByVal Align As Long)
    Var nStyle As Long
    Var Ret As Long

    Align = Check1.GetCheck

    nStyle = Api_GetWindowLong(List1.GethWnd, GWL_EXSTYLE)

    Select Case Align
        Case 0
            nStyle = nStyle And Not WS_EX_LEFTSCROLLBAR
        Case 1
            nStyle = nStyle Or WS_EX_LEFTSCROLLBAR
    End Select

    Ret = Api_SetWindowLong(List1.GethWnd, GWL_EXSTYLE, nStyle)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
   Var x As Long

   For x = 0 To 20
      List1.AddString Format$(Rnd(1) * 1000000, "#######.#####")
   Next

   Check1.SetWindowText "スクロールバー[左]"
   Check2.SetWindowtext "スクロールバー[右]"
End Sub

'=====================================
'=
'=====================================
Declare Sub Check1_on edecl ()
Sub Check1_on()
    Check2.SetCheck 0
    ListScrollAlign(Align)
End Sub

'=====================================
'=
'=====================================
Declare Sub Check2_on edecl ()
Sub Check2_on()
    Check1.SetCheck 0
    ListScrollAlign(Align)
End Sub

'=====================================
'=
'=====================================
While 1
```

```
    WaitEvent
Wend
Stop
End
```

スクロールバーを表示する

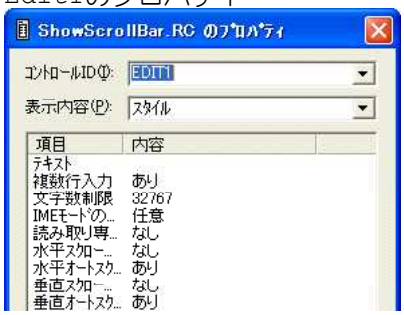
ShowScrollBar スクロールバーを表示

Edit1のプロパティは最下段参照
水平スクロールバーを選択した場合

垂直スクロールバーを選択した場合

水平・垂直両方のスクロールバーを選択した場合

Edit1のプロパティ　　複数行入力『なし』の場合



複数行入力『なし』の場合スクロールバーは表示されますが変化しません。

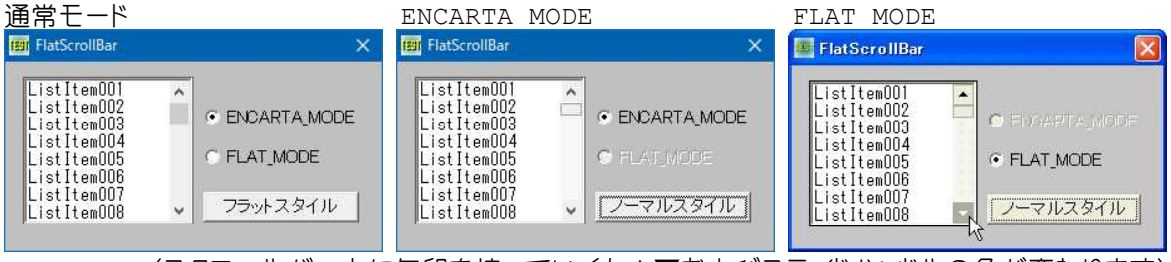
```
'=======================================
'= スクロールバーを表示する
'=    (ShowScrollBar2.bas)
'=======================================
#include "Windows.bi"

' スクロールバーを表示
Declare Function Api_ShowScrollBar& Lib "user32" Alias "ShowScrollBar" (ByVal hWnd&,
```

```
ByVal wBar&, ByVal bShow&)
#define SB_BOTH 3                                   '標準スクロールバーの水平･垂直両方
#define SB_CTL 2                                    'スクロールバーコントロールを指定する
#define SB_HORZ 0                                   '標準水平スクロールバーを指定する
#define SB_SETPARTS &H404                           'WM_USER + 4
#define SB_SETTEXTA &H401                           'WM_USER + 1
#define SB_VERT 1                                   '標準垂直スクロールバーを指定する

Var Shared Edit1 As Object
Var Shared Radio(2) As Object
For i = 0 To 2
    Radio(i).Attach GetDlgItem("Radio" & Trim$(Str$(i + 1)))
    Radio(i).SetFontSize 14
Next
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Function Index bdecl () As Integer
Function Index()
    Index = Val(Mid$(GetDlgRadioSelect("Radio1"), 6)) - 1
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    Select Case Index
        Case 0
            sbType = SB_HORZ
            Radio(1).EnableWindow 0
            Radio(2).EnableWindow 0
        Case 1
            sbType = SB_VERT
            Radio(0).EnableWindow 0
            Radio(2).EnableWindow 0
        Case 2
            sbType = SB_BOTH
            Radio(0).EnableWindow 0
            Radio(1).EnableWindow 0
    End Select

    Ret = Api_ShowScrollBar(Edit1.GethWnd, sbType, True)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

スクロールバーをフラットスタイルに(Ⅰ)

リストボックス内のスクロールバーをフラット(ENCARTA_MODE･FLAT_MODE)に。
FlatSB_SetScrollPos フラットスクロールバーの位置を設定
FlatSB_SetScrollProp フラットスクロールバーのInitializeFlatSBが呼ばれたかどうかを設定
InitializeFlatSB フラットスクロールバーを初期化
UninitializeFlatSB フラットスクロールバーを初期化しない(標準スクロールバーに戻す)

通常モード　　ENCARTA_MODE　　FLAT_MODE



FLAT_MODE(スクロールバー上に矢印を持っていくと▲▼およびスライドハンドルの色が変わります)

```
'=====================================
'= スクロールバーをフラットスタイルに(Ⅰ)
'=    (FlatScrollBar.bas)
'=====================================
#include "Windows.bi"

' フラットスクロールバーの位置を設定
Declare Function Api_FlatSB_SetScrollPos& Lib "Comctl32" Alias "FlatSB_SetScrollPos"
(ByVal hWnd&, ByVal code&, ByVal nPos&, ByVal fRedraw&)

' フラットスクロールバーのInitializeFlatSBが呼ばれたかどうかを設定
Declare Function Api_FlatSB_SetScrollProp& Lib "Comctl32" Alias "FlatSB_SetScrollProp"
(ByVal hWnd&, ByVal index&, ByVal newValue&, ByVal fRedraw&)

' フラットスクロールバーを初期化
Declare Function Api_InitializeFlatSB& Lib "Comctl32" Alias "InitializeFlatSB" (ByVal
hWnd&)

' フラットスクロールバーを初期化しない(標準スクロールバーに戻す)
Declare Function Api_UninitializeFlatSB& Lib "Comctl32" Alias "UninitializeFlatSB"
(ByVal hWnd&)

#define WSB_PROP_VSTYLE &H100              '垂直スクロールバーの外観を変える

#define FSB_ENCARTA_MODE 1
#define FSB_FLAT_MODE 2
#define FSB_REGULAR_MODE 0

#define SB_VERT 1

Var Shared Mainform As Object
Var Shared List1 As Object
Var Shared Button1 As Object
Var Shared Radio1 As Object
Var Shared Radio2 As Object
Var Shared Timer1 As Object

Mainform.Attach GethWnd
List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
Radio1.Attach GetDlgItem("Radio1") : Radio1.SetFontSize 14
Radio2.Attach GetDlgItem("Radio2") : Radio2.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Timer1.Attach GetDlgItem("Timer1")

Var Shared Flag As Integer
Var Shared sFlag As Integer

'=====================================
'=
'=====================================
Declare Sub Mainform_Start edecl ()
Sub Mainform_Start()
    Var i As Integer

    Flag = 1
    sFlag = 0

    List1.Resetcontent
    For i = 1 To 100
```

```
        List1.AddString "ListItem" & Right$(Str$(1000 + i),3)
    Next

    Timer1.SetInterval 10
    Timer1.Enable -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    If Flag = 0 Then
        Ret = Api_UninitializeFlatSB(List1.GethWnd)
        Button1.SetWindowText "フラットスタイル"
        Radio1.EnableWindow -1
        Radio2.EnableWindow -1
    Else
        Ret = Api_InitializeFlatSB(List1.GethWnd)
        If sFlag = 0 Then
            Ret = Api_FlatSB_SetScrollProp(List1.GethWnd, WSB_PROP_VSTYLE,
FSB_ENCARTA_MODE, True)
        Else
            Ret = Api_FlatSB_SetScrollProp(List1.GethWnd, WSB_PROP_VSTYLE,
FSB_FLAT_MODE, True)
        End If

        Button1.SetWindowText "ノーマルスタイル"
        If Radio1.GetCheck = 0 Then Radio1.EnableWindow 0
        If Radio2.GetCheck = 0 Then Radio2.EnableWindow 0
    End If

    Flag = Flag - 1
    Flag = Abs(Flag)
End Sub

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_timer()
    Var TopIdx As Long
    Var Ret As Long

    TopIdx = List1.GetTopIndex + 1
    Ret = Api_FlatSB_SetScrollPos(List1.GethWnd, SB_VERT, TopIdx, True)
End Sub

'=====================================
'=
'=====================================
Declare Sub Radio1_on edecl ()
Sub Radio1_on()
    sFlag = 0
End Sub

'=====================================
'=
'=====================================
Declare Sub Radio2_on edecl ()
Sub Radio2_on()
    sFlag = 1
End Sub

'=====================================
'=
'=====================================
Declare Sub Mainform_QueryClose edecl (Cancel%,ByVal Mode%)
```

```
Sub Mainform_QueryClose(Cancel%,ByVal Mode%)
    Var Ret As Long

    Ret = Api_UninitializeFlatSB(List1.GethWnd)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

スクロールバーをフラットスタイルに(II)

フォームにフラット垂直スクロールバーをセットします。
GetWindowLong 指定されたウィンドウに関しての情報を取得
SetWindowLong 指定されたウィンドウの属性を変更
InitializeFlatSB フラットスクロールバーを初期化
UninitializeFlatSB フラットスクロールバーを未初期化(標準に戻す)
FlatSB_ShowScrollBar フラットスクロールバーを表示
FlatSB_EnableScrollBar フラットスクロールバーの使用可・使用不可にする
FlatSB_GetScrollProp フラットスクロールバーのInitializeFlatSBが呼ばれたかどうかを取得
FlatSB_SetScrollProp フラットスクロールバーのInitializeFlatSBが呼ばれたかどうかを設定
FlatSB_GetScrollInfo フラットスクロールバーの情報を取得
FlatSB_SetScrollInfo フラットスクロールバーの情報を設定
FlatSB_GetScrollRange フラットスクロールバーの移動範囲を取得
FlatSB_SetScrollRange フラットスクロールバーの移動範囲を設定
FlatSB_GetScrollPos フラットスクロールバーの位置を取得
FlatSB_SetScrollPos フラットスクロールバーの位置を設定
GetSysColor システムの背景色を取得

例ではUPボタンを使用不可にしています。


```
'=====================================
'= フラットスクロールバー作成(II)
'=    (FlatScrollBar3.bas)
'=====================================
#include "Windows.bi"

#define WS_VSCROLL &H200000                 '垂直スクロールバーを持つウィンドウを作成する
#define WS_HSCROLL &H100000                 '水平スクロールバーを持つウィンドウを作成する
#define GWL_STYLE -16                       'アプリケーションのインスタンスハンドル

#define WSB_PROP_CXHSCROLL &H2              '水平スクロールバーの方向ボタンの幅
#define WSB_PROP_CXHTHUMB &H10              '水平スクロールバーのスクロールつまみの幅
#define WSB_PROP_CXVSCROLL &H8              '垂直スクロールバーの幅
#define WSB_PROP_CYHSCROLL &H4              '水平スクロールバーの高さ
#define WSB_PROP_CYVSCROLL &H1              '垂直スクロールバーの方向ボタンの高さ
#define WSB_PROP_CYVTHUMB &H20              '垂直スクロールバーのスクロールつまみの高さ
#define WSB_PROP_HBKGCOLOR &H80             '水平スクロールバーの背景色を表す(COLORREF値)
#define WSB_PROP_HSTYLE &H200               '水平スクロールバーの外観を変える
#define WSB_PROP_MASK &HFFF                 '
#define WSB_PROP_PALETTE &H800              'スクロールバーが描画される時に使われるHPALETTE値
#define WSB_PROP_VBKGCOLOR &H40             '垂直スクロールバーの背景色を表す(COLORREF値)
#define WSB_PROP_VSTYLE &H100               '垂直スクロールバーの外観を変える
#define WSB_PROP_WINSTYLE &H400             '
```

```
#define FSB_REGULAR_MODE 0
#define FSB_ENCARTA_MODE 1
#define FSB_FLAT_MODE 2

#define SB_HORZ 0                                   '標準水平スクロールバーを指定
#define SB_VERT 1                                   '標準垂直スクロールバーを指定
#define SB_CTL 2                                    'スクロールバーコントロールを指定
#define SB_BOTH 3                                   '標準スクロールバーの水平･垂直両方

' スクロールバー矢印の使用可能不可能を設定することを示す定数
#define ESB_ENABLE_BOTH &H0                         '両方の矢印ボタンを有効にする
#define ESB_DISABLE_LEFT &H1                        '水平スクロールバーの左方向の矢印ボタンを無効にする
#define ESB_DISABLE_RIGHT &H2                       '水平スクロールバーの右方向の矢印ボタンを無効にする
#define ESB_DISABLE_BOTH &H3                        '両方の矢印ボタンを無効にする
#define ESB_DISABLE_UP &H1                          '垂直スクロールバーの上方向の矢印ボタンを無効にする
#define ESB_DISABLE_DOWN &H2                        '垂直スクロールバーの下方向の矢印ボタンを無効にする
#define ESB_DISABLE_LTUP &H1                        '(ESB_DISABLE_LEFT)
#define ESB_DISABLE_RTDN &H2                        '(ESB_DISABLE_RIGHT)

#define SIF_RANGE &H1                               'nMinとnMaxの設定を明示
#define SIF_PAGE &H2                                'nPageの設定を明示
#define SIF_POS &H4                                 'nPosの設定を明示
#define SIF_ALL &H7                                 '(SIF_RANGE Or SIF_PAGE Or SIF_POS)

Type SCROLLINFO
    cbSize      As Long                             'このレコードのバイトサイズ
    fMask       As Long                             '取得･設定する値を指定するマスクフラグ
    nMin        As Long                             'スクロール領域の最小値
    nMax        As Long                             'スクロール領域の最大値
    nPage       As Long                             'サム(つまみ)のサイズ
    nPos        As Long                             'サムの位置
    nTrackPos   As Long                             'ドラッグ中のサムの位置
End Type

' 指定されたウィンドウに関しての情報を取得。また、拡張ウィンドウメモリから、指定されたオフセットにある32ビット値を取得することもできる
Declare Function Api_GetWindowLong& Lib "user32" Alias "GetWindowLongA" (ByVal hWnd&,
ByVal nIndex&)

' 指定されたウィンドウの属性を変更。また、拡張ウィンドウメモリの指定されたオフセットの32ビット値を書き換えることができる
Declare Function Api_SetWindowLong& Lib "user32" Alias "SetWindowLongA" (ByVal hWnd&,
ByVal nIndex&, ByVal dwNewLong&)

' フラットスクロールバーを初期化
Declare Function Api_InitializeFlatSB& Lib "comctl32" Alias "InitializeFlatSB" (ByVal
hWnd&)

' フラットスクロールバーを未初期化(標準スクロールバーに戻す)
Declare Function Api_UninitializeFlatSB& Lib "comctl32" Alias "UninitializeFlatSB"
(ByVal hWnd&)

' フラットスクロールバーの表示
Declare Function Api_FlatSB_ShowScrollBar& Lib "comctl32" Alias "FlatSB_ShowScrollBar"
(ByVal hWnd&, ByVal code&, ByVal fShow&)

' フラットスクロールバーを使用可･使用不可にする
Declare Function Api_FlatSB_EnableScrollBar& Lib "comctl32" Alias
"FlatSB_EnableScrollBar" (ByVal hWnd&, ByVal wSBFlags&, ByVal wArrows&)

' フラットスクロールバーのInitializeFlatSBが呼ばれたかどうかを取得
Declare Function Api_FlatSB_GetScrollProp& Lib "comctl32" Alias "FlatSB_GetScrollProp"
(ByVal hWnd&, ByVal propIndex&, ByVal pValue&)

' フラットスクロールバーのInitializeFlatSBが呼ばれたかどうかを設定
Declare Function Api_FlatSB_SetScrollProp& Lib "comctl32" Alias "FlatSB_SetScrollProp"
(ByVal hWnd&, ByVal index&, ByVal newValue&, ByVal fRedraw&)

' フラットスクロールバーの情報を取得
Declare Function Api_FlatSB_GetScrollInfo& Lib "comctl32" Alias "FlatSB_GetScrollInfo"
```

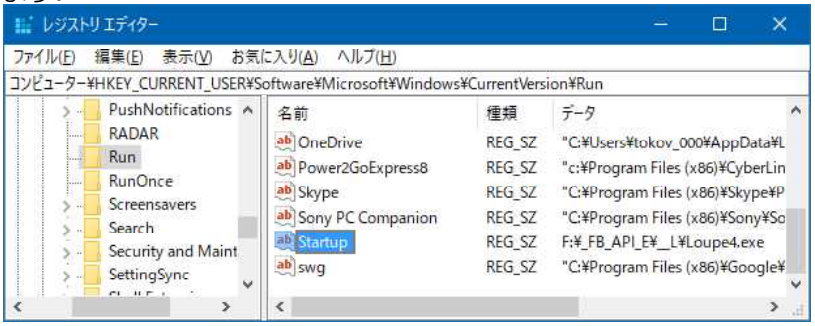
```
(ByVal hWnd&, ByVal code&, lpsi As Any)

' フラットスクロールバーの情報を設定
Declare Function Api_FlatSB_SetScrollInfo& Lib "comctl32" Alias "FlatSB_SetScrollInfo"
(ByVal hWnd&, ByVal code&, lpsi As Any, ByVal fRedraw&)

' フラットスクロールバーの移動範囲を取得
Declare Function Api_FlatSB_GetScrollRange& Lib "comctl32" Alias
"FlatSB_GetScrollRange" (ByVal hWnd&, ByVal code&, ByVal lpMinPos&, ByVal lpMaxPos&)

' フラットスクロールバーの移動範囲を設定
Declare Function Api_FlatSB_SetScrollRange& Lib "comctl32" Alias
"FlatSB_SetScrollRange" (ByVal hWnd&, ByVal code&, ByVal nMinPos&, ByVal nMaxPos&, ByVal
fRedraw&)

' フラットスクロールバーの位置を取得
Declare Function Api_FlatSB_GetScrollPos& Lib "comctl32" Alias "FlatSB_GetScrollPos"
(ByVal hWnd&, ByVal code&)

' フラットスクロールバーの位置を設定
Declare Function Api_FlatSB_SetScrollPos& Lib "comctl32" Alias "FlatSB_SetScrollPos"
(ByVal hWnd&, ByVal code&, ByVal nPos&, ByVal fRedraw&)

' システムの背景色を取得
Declare Function Api_GetSysColor& Lib "user32" Alias "GetSysColor" (ByVal nIndex&)

' nIndexで指定する定数
#define COLOR_BTNFACE 15                    'コマンドボタンの表面色

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var rgbColor As Long
    Var SI As SCROLLINFO
    Var Min As Long
    Var Max As Long
    Var Ret As Long

    'Buttonの表面色を取得(EDE9EC)
    rgbColor = Api_GetSysColor(COLOR_BTNFACE)

    'Mainformを取得色で塗る
    SetBackColor rgbColor

    '画面を消去
    Cls

    'Mainformを表示
    ShowWindow -1

    'フォームにスクロールバーをセット
    Ret = Api_GetWindowLong(GethWnd, GWL_STYLE)
    Ret = Ret Or WS_VSCROLL Or WS_HSCROLL
    Ret = Api_SetWindowLong(GethWnd, GWL_STYLE, Ret)

    '初期化
    Ret = Api_InitializeFlatSB(GethWnd)

    '垂直スクロールバーをEncarta-modeに
    Ret = Api_FlatSB_SetScrollProp(GethWnd, WSB_PROP_VSTYLE, FSB_ENCARTA_MODE, False)

    '垂直スクロールバーからボタンアップを使用不可に
    Ret = Api_FlatSB_EnableScrollBar(GethWnd, SB_VERT, ESB_DISABLE_UP)

    'スクロールレンジセット
    Ret = Api_FlatSB_SetScrollRange(GethWnd, SB_VERT, 0, 80, False)
```

```
    'スクロールポジション設定
    Ret = Api_FlatSB_SetScrollPos(GethWnd, SB_VERT, 50, False)

    '水平スクロールバーを消す
    Ret = Api_FlatSB_ShowScrollBar(GethWnd, SB_HORZ, False)

    'スクロールバー情報取得
    SI.cbSize = Len(SI)
    SI.fMask = SIF_ALL
    Ret = Api_FlatSB_GetScrollInfo(GethWnd, SB_VERT, SI)

    SI.nPos = SI.nPos - 10

    '新しいスクロールバー情報を設定
    Ret = Api_FlatSB_SetScrollInfo(GethWnd, SB_VERT, SI, True)

    Ret = Api_FlatSB_GetScrollRange(GethWnd, SB_VERT, Min, Max)
    Symbol(20, 40), "Scroll Position:" & Str$(Int(100 *
(Api_FlatSB_GetScrollPos(GethWnd, SB_VERT) / Max))) & "%", 1, 1
End Sub

'=======================================
'=
'=======================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
    Var Ret As Long

    'フラットスクロールバーの削除
    Ret = Api_UninitializeFlatSB(GethWnd)
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

スタートアップにEXEを登録

Windows起動時に起動させるEXEをレジストリに登録します。
RegCreateKeyEx レジストリキーを作成
RegSetValueEx レジストリキーの値を設定
RegCloseKey レジストリのハンドルを解放
FormatMessage メッセージの文字列を指定の書式で取得

「開く」で、スタートアップに登録するEXEを選択し、「登録実行」でレジストリに書き込みます。
レジストリエディタで確認をしています。




```
'=======================================
'= スタートアップにEXEを登録
'=    (SetStartup.bas)
'=======================================
```

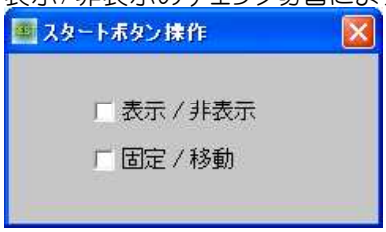
```
#include "Windows.bi"

Type SECURITY_ATTRIBUTES
    nLength                 As Long
    lpSecurityDescriptor    As Long
    bInheritHandle          As Long
End Type

' レジストリキーを作成
Declare Function Api_RegCreateKeyEx& Lib "advapi32" Alias "RegCreateKeyExA" (ByVal
hKey&, ByVal lpSubKey$, ByVal Reserved&, ByVal lpClass$, ByVal dwOptions&, ByVal
samDesired&, lpSecurityAttributes As SECURITY_ATTRIBUTES, phkResult&, lpdwDisposition&)

' レジストリキーの値を設定
Declare Function Api_RegSetValueEx& Lib "advapi32" Alias "RegSetValueExA" (ByVal hKey&,
ByVal lpValueName$, ByVal Reserved&, ByVal dwType&, lpData As Any, ByVal cbData&)

' レジストリのハンドルを解放
Declare Function Api_RegCloseKey& Lib "advapi32" Alias "RegCloseKey" (ByVal hKey&)

' メッセージの文字列を指定の書式で取得
Declare Function Api_FormatMessage& Lib "Kernel32" Alias "FormatMessageA" (ByVal
dwFlags&, lpSource As Any, ByVal dwMessageId&, ByVal dwLanguageId&, ByVal lpBuffer$,
ByVal nSize&, Arguments&)

#define ERROR_SUCCESS &H0                   '正常終了の戻り値を示す
#define HKEY_CURRENT_USER -2147483647       '現在Windowsにログインしているユーザーの情報
#define REG_OPTION_NON_VOLATILE 0           '設定内容をレジストリに保存
#define REG_BINARY 3                        'バイナリデータ
#define REG_DWORD 4                         '32ビットの数値
#define REG_DWORD_BIG_ENDIAN 5              'ビッグエンディアン形式の32ビット数値
#define REG_DWORD_LITTLE_ENDIAN 4           'リトルエンディアン形式の32ビット数値(REG_DWORDと同
                                            等)
#define REG_EXPAND_SZ 2                     '展開前の環境変数への参照が入ったヌル終端文字列
#define REG_LINK 6                          'Unicodeシンボリックリンク
#define REG_MULTI_SZ 7                      'ヌル終端文字列の配列
#define REG_QWORD 11                        '64ビット数値
#define REG_QWORD_LITTLE_ENDIAN 11          'リトルエンディアン形式の64ビット数値(REG_QWORDと同
                                            等)
#define REG_RESOURCE_LIST 8                 'デバイスドライバのリソースリスト
#define REG_SZ 1                            'ヌル終端文字列
#define KEY_SET_VALUE &H2                   'レジストリの値を設定する
#define FORMAT_MESSAGE_FROM_SYSTEM &H1000 'システムメッセージリソースを検索する

Var Shared Edit1 As Object
Var Shared Button1 As Object
Var Shared Button2 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14

Var Shared FileName As String

'====================================
'= Startupに設定するファイル名
'====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    FileName = WinOpenDlg("ファイルのオープン", "*.exe", "アプリケーション(*.exe)", 0)
    If FileName <> Chr$(&H1B) Then
        Edit1.SetWindowText FileName
    End If
End Sub

'====================================
'=
'====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
```

```
    Var sa As SECURITY_ATTRIBUTES
    Var SubKey As String
    Var MSG As String
    Var sError As String
    Var Result As Long
    Var Disposition As Long
    Var Ret As Long

    SubKey = "Software¥Microsoft¥Windows¥CurrentVersion¥Run"

    'レジストリにキーを作成
    Ret = Api_RegCreateKeyEx(HKEY_CURRENT_USER, SubKey, 0, ByVal 0,
REG_OPTION_NON_VOLATILE, KEY_SET_VALUE, sa, Result, Disposition)

    If Ret = ERROR_SUCCESS Then
        FileName = Edit1.GetWindowText
        If FileName = "" Then EXit Sub
        a = 0
        For b = 1 To Len(FileName)
            c = Asc(Mid$(FileName, b, 1))
            If c > &H39 Or c < &H30 Then a = 1 : Exit For
        Next

        If a = 1 Then
            Ret = Api_RegSetValueEx(Result, "Startup", 0, REG_SZ, FileName, Len(FileName)
+ 1)
        End If

        If Ret <> 0 Then

            'エラーコードからエラーメッセージを取得
            sError = String$(260, Chr$(0))
            Ret = Api_FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM, ByVal 0, Ret, 0, sError,
Len(sError), 0)
            A% = MessageBox(GetWindowText, "エラー！", 0, 2)
        Else
            MSG = MSG & "「" & SubKey & "」に" & Chr$(13, 10) & Chr$(13, 10)
            MSG = MSG & FileName & " を登録しました！"
            A% = MessageBox(GetWindowText, MSG, 0, 2)
        End If
    Else

        'エラーコードからエラーメッセージを取得
        sError = String$(260, Chr$(0))
        Ret = Api_FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM, ByVal 0, hKey, 0, sError,
Len(sError), 0)
        A% = MessageBox(GetWindowText, sError, 0, 2)
    End If
End Sub

'====================================
'=
'====================================
While 1
    WaitEvent
Wend
Stop
End
```

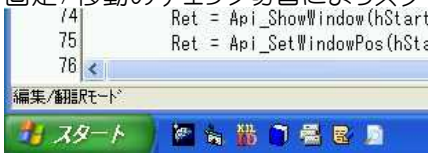
スタートボタンの操作

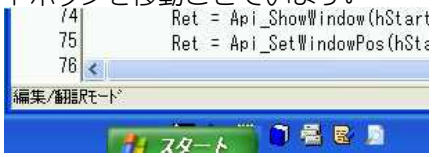
スタートボタンの表示･非表示とスタートボタンを移動させてみます。
FindWindowEx クラス名 、または キャプションを与えてウィンドウのハンドルを取得
SetWindowPos ウィンドウのサイズ、位置、および Z オーダーを設定
SendMessage ウィンドウにメッセージを送信
ShowWindow 指定されたウィンドウの表示状態を設定

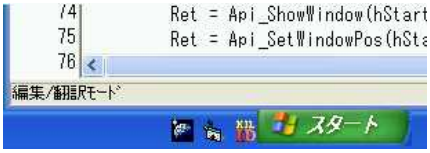
表示/非表示のチェック切替によりスタートボタンを表示/非表示させています。



固定/移動のチェック切替によりスタートボタンを移動させています。








```
'=====================================
'= スタートボタンの操作
'=    (SetWindowPos6.bas)
'=====================================
#include "Windows.bi"

' クラス名 、または キャプションを与えてウィンドウのハンドルを取得
Declare Function Api_FindWindowEx& Lib "user32" Alias "FindWindowExA" (ByVal
hWndParent&, ByVal hWndChildAfter&, ByVal lpszClassName As Any, ByVal lpszWindow As Any)

' ウィンドウのサイズ、位置、および Z オーダーを設定
Declare Function Api_SetWindowPos& Lib "user32" Alias "SetWindowPos" (ByVal hWnd&, ByVal
hWndInsertAfter&, ByVal X&, ByVal Y&, ByVal CX&, ByVal CY&, ByVal uFlags&)

' ウィンドウにメッセージを送信。この関数は、指定したウィンドウのウィンドウプロシージャが処理を終了するまで制御を返さない
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, ByVal lParam&)

' 指定されたウィンドウの表示状態を設定
Declare Function Api_ShowWindow& Lib "user32" Alias "ShowWindow" (ByVal hWnd&, ByVal
nCmdShow&)

' 画像ファイルの読み込み
Declare Function Api_LoadImage& Lib "user32" Alias "LoadImageA" (ByVal hInst&, ByVal
lpszName$, ByVal uType&, ByVal cxDesired&, ByVal cyDesired&, ByVal fuLoad&)

#define SW_HIDE 0                          '指定のウィンドウを非表示にし他のウィンドウをアクティブ化
#define SW_RESTORE 9                       'ウィンドウをアクティブ化し表示。ウィンドウがアイコン化または最大化されているときは元の位置とサイズに

#define GW_CHILD 5                         '基準となるウィンドウの子ウィンドウのうちトップレベルのウィンドウを検索

#define SWP_NOSIZE &H1                     'ウィンドウの現在のサイズを保持する
#define WM_CLOSE &H10                      'ウィンドウ或いはアプリケーションをクローズされた
#define BM_GETIMAGE &HF6                   'ボタンに関連付けられているイメージのハンドルを取得
#define BM_SETIMAGE &HF7                   'イメージをボタンに関連付ける
#define IMAGE_BITMAP 0                     'ビットマップ
#define LR_LOADFROMFILE &H10               '外部ファイルからロードする

Var Shared Check1 As Object
Var Shared Check2 As Object
Var Shared Timer1 As Object

Check1.Attach GetDlgItem("Check1") : Check1.SetFontSize 14
Check2.Attach GetDlgItem("Check2") : Check2.SetFontSize 14
```

```
Timer1.Attach GetDlgItem("Timer1")
Var Shared hStart As Long
Var Shared hOldPic As Long
Var Shared X As Long
Var Shared Y As Long
Var Shared stp As Long

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var Ret As Long

    X = 0
    stp = 2

    Ret = Api_FindWindowEx(0, 0, "Shell_TrayWnd", 0)
    hStart = Api_FindWindowEx(Ret, 0, "BUTTON", 0)

    hOldPic = Api_SendMessage(hStart, BM_GETIMAGE, IMAGE_BITMAP, ByVal 0)

    Timer1.SetInterval 10
    Timer1.Enable 0
End Sub

'=====================================
'= スタートボタンの表示/非表示
'=====================================
Declare Sub Check1_on edecl ()
Sub Check1_on()
    Var Ret As Long

    If Check1.GetCheck = 1 Then
        Ret = Api_ShowWindow(hStart, SW_HIDE)
    Else
        Ret = Api_ShowWindow(hStart, SW_RESTORE)
        Ret = Api_SetWindowPos(hStart, 0, 0, 0, 0, 0, SWP_NOSIZE)
    End If
End Sub

'=====================================
'= タイマーのOn/Off
'=====================================
Declare Sub Check2_on edecl ()
Sub Check2_on()
    If Check2.GetCheck = 1 Then
        Timer1.Enable -1
    Else
        Timer1.Enable 0
    End If
End Sub

'=====================================
'= スタートボタンを移動させる
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var Ret As Long

    Y = sin(X / 10) * 10

    Ret = Api_SetWindowPos(hStart, 0, X, Y, 0, 0, SWP_NOSIZE)

    X = X + stp

    If X > 200 Then stp = -2
    If X < 0 Then stp = 2
End Sub
```

```
'=====================================
'= スタートボタンを所定の位置へ
'=====================================
Declare Sub ResetStartButton()
Sub ResetStartButton()
    Var Ret As Long

    Timer1.Enable 0

    Ret = Api_ShowWindow(hStart, SW_RESTORE)
    Ret = Api_SetWindowPos(hStart, 0, 0, 0, 0, 0, SWP_NOSIZE)
    Ret = Api_SendMessage(hStart, BM_SETIMAGE, IMAGE_BITMAP, ByVal hOldPic)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()

    ResetStartButton
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

スタートメニューを開く

MapVirtualKey 仮想キーコード･ASCII値･スキャンコード間でコードを変換
keybd_event 特殊キーの状態を設定

上下に分けて表示しています

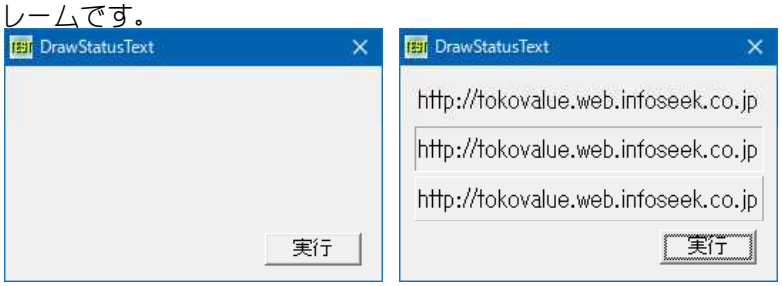

```
'=====================================
'= スタートメニューを開く
'=    (MapVirtualKey.bas)
'=====================================
#include "Windows.bi"

' 仮想キーコード･ASCII値･スキャンコード間でコードを変換
Declare Function Api_MapVirtualKey& Lib "user32" Alias "MapVirtualKeyA" (ByVal wCode&,
ByVal wMapType&)

' 特殊キーの状態を設定
Declare Sub Api_keybd_event Lib "user32" Alias "keybd_event" (ByVal bVk As byte, ByVal
bScan As byte, ByVal dwFlags&, ByVal dwExtraInfo&)

#define KEYEVENTF_KEYUP &H2                    'キーを放す
#define MENU_KEYCODE 91
```

```
'====================================
'=
'====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Api_keybd_event MENU_KEYCODE, 0, 0, 0
    Api_keybd_event MENU_KEYCODE, 0, KEYEVENTF_KEYUP, 0
End Sub

'====================================
'=
'====================================
While 1
    WaitEvent
Wend
Stop
End
```

ステータスバースタイルでテキストを表示

DrawStatusText ステータスバースタイルでテキストを表示
GetDC デバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放
GetSysColor システムの背景色を取得

ステータスバースタイル(フレーム)でテキストを表示します。1行目はフラット、2行目は凹型フレーム、3行目は凸型フレームです。




```
'====================================
'= ステータスバースタイルでテキストを表示
'=    (DrawStatusText.bas)
'====================================
#include "Windows.bi"

Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

' ステータスバースタイルでテキストを表示
Declare Sub Api_DrawStatusText Lib "comctl32" Alias "DrawStatusTextA" (ByVal hDC&,
DrawRect As RECT, ByVal Text$, ByVal Flags&)

' RECT構造体の座標値を拡大・縮小
Declare Sub Api_InflateRect Lib "user32" Alias "InflateRect" (lpRect As RECT, ByVal x As
Long, ByVal y As Long)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

' システムの背景色を取得
Declare Function Api_GetSysColor& Lib "user32" Alias "GetSysColor" (ByVal nIndex&)
```

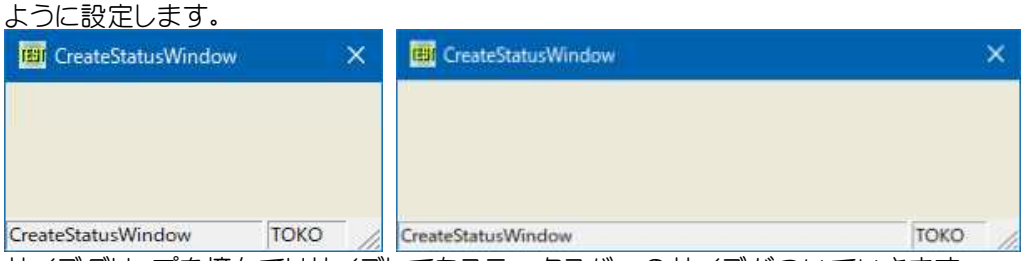
```
#define COLOR_BTNFACE 15                    '3Dオブジェクトの表面色
#define SBT_NOBORDERS &H100                 '枠線なし
#define SBT_POPOUT &H200                    '凸型枠
#define SBT_RTLREADING &H400                'テキストは右から左に表示
#define SBT_SUNKEN &H0                      '凹型枠(デフォルト)

Var Shared Button1 As Object

Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var rgbColor As Long

    'Buttonの表面色を取得(EDE9EC)
    rgbColor = Api_GetSysColor(COLOR_BTNFACE)

    'Mainformを取得色で塗り
    SetBackColor rgbColor

    '画面を消去し
    cls

    'Mainformを表示
    ShowWindow -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var rc As RECT
    Var hDC As Long
    Var txt As String
    Var Ret As Long

    txt = "http://tokovalue.web.infoseek.co.jp"
    hDC = Api_GetDC(GethWnd)

    rc.Left = 10
    rc.Top = 8
    rc.Right = 255
    rc.Bottom = 40

    'フラット
    Api_DrawStatusText hDC, rc, txt, SBT_NOBORDERS

    rc.Top = 43
    rc.Bottom = 75

    '凹型フレーム
    Api_DrawStatusText hDC, rc, txt, SBT_SUNKEN

    rc.Top = 78
    rc.Bottom = 110

    '凸型フレーム
    Api_DrawStatusText hDC, rc, txt, SBT_POPOUT

    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

'=====================================
'=
'=====================================
While 1
```

```
    WaitEvent
Wend
Stop
End
```

ステータスバーの作成(Ⅰ)

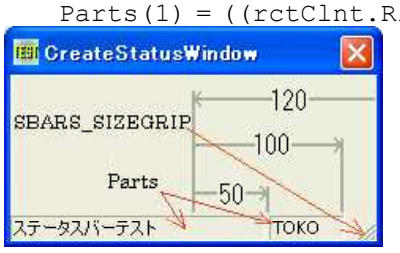
ステータスバーを作成します。
InitCommonControls コモンコントロールの初期化
CreateStatusWindow ステータスバーの作成
GetClientRect ウィンドウのクライアント領域の座標を取得
MoveWindow ウィンドウの位置とサイズを変更
DestroyWindow ウィンドウの破棄
SendMessage (Api_SendMessageByAny)パーツの個数設定
SendMessage (Api_SendMessageByString)パーツにテキストを設定

メインフォームは可視なしに設定し、フォーム表面色を設定後表示しています。メインフォームはサイズの変更ができるように設定します。



サイズグリップを摘んでリサイズしてもステータスバーのサイズがついていきます。
矩形位置の計算

```
    Parts(0) = ((rctClnt.Right - rctSBar.Left) - 120) + 50
    Parts(1) = ((rctClnt.Right - rctSBar.Left) - 120) + 100
```




```
'=======================================
'= ステータスバーを作成(Ⅰ)
'=    (CreateStatusWindow2.bas)
'=======================================
#include "Windows.bi"

Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

'コモンコントロールの初期化
Declare Sub Api_InitCommonControls Lib "Comctl32" Alias "InitCommonControls" ()

'ステータスウィンドウの作成
Declare Function Api_CreateStatusWindow& Lib "Comctl32" Alias "CreateStatusWindowA"
(ByVal Style&, ByVal lpszText$, ByVal hWndParent&, ByVal wID&)

' ウィンドウのクライアント領域の座標を取得
Declare Function Api_GetClientRect& Lib "user32" Alias "GetClientRect" (ByVal hWnd&,
lpRect As RECT)

' 指定されたウィンドウの位置およびサイズを変更
Declare Function Api_MoveWindow& Lib "user32" Alias "MoveWindow" (ByVal hWnd&, ByVal x&,
ByVal y&, ByVal nWidth&, ByVal nHeight&, ByVal bRepaint&)
```

```
' CreateWindowExの解放
Declare Function Api_DestroyWindow& Lib "user32" Alias "DestroyWindow" (ByVal hWnd&)

' ウィンドウにメッセージを送信
Declare Function Api_SendMessageByAny& Lib "user32" Alias "SendMessageA" (ByVal hWnd&,
ByVal wMsg&, ByVal wParam&, lParam As Any)
Declare Function Api_SendMessageByString& Lib "user32" Alias "SendMessageA" (ByVal
hWnd&, ByVal wMsg&, ByVal wParam&, ByVal lParam$)

#define WM_USER &H400                          'ユーザーが定義できるメッセージの使用領域を表すだけで
                                               これ自体に意味はない
#define WS_CHILD &H40000000                    '親ウインドウを持つコントロール(子ウインドウ)を作成する
#define WS_VISIBLE &H10000000                  '可視状態のウィンドウを作成する
#define WS_BORDER &H800000                     'フォームの枠線がある
#define SBARS_SIZEGRIP &H100                   'サイズグリップ
#define CCS_BOTTOM &H3                         'ウィンドウを親ウィンドウの下端に配置
#define SB_SETTEXTA &H401                      'WM_USER + 1
#define SB_SETPARTS &H404                      'WM_USER + 4

Var Shared SBhWnd As Long
Var Shared lpszTitle$ As String
'=======================================
'=
'=======================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var rgbColor As Long

    '表面色を設定
    rgbColor = RGB(236, 233, 216)

    'MainFormを取得色で塗り
    SetBackColor rgbColor

    '画面を消去し
    Cls

    'MainFormを表示
    ShowWindow -1

    Api_InitCommonControls

    'ステータスバーの追加
    SBhWnd = Api_CreateStatusWindow(WS_CHILD Or WS_VISIBLE Or WS_BORDER Or CCS_BOTTOM Or
SBARS_SIZEGRIP, lpszTitle$, GethWnd, 101)
End Sub

'=======================================
'=
'=======================================
Declare Sub MainForm_Resize edecl ()
Sub MainForm_Resize()
    Var rctSBar As RECT                        'スクロールバー矩形
    Var rctClnt As RECT                        'フォーム矩形
    Var Parts(1) As Long                       'スクロールバー部品数
    Var Res As Long

    'メインフォームの矩形を取得
    Res = Api_GetClientRect(GethWnd, rctClnt)

    'ステータスバーウィンドウの矩形を取得
    Res = Api_GetClientRect(SBhWnd, rctSBar)

    Parts(0) = ((rctClnt.Right - rctSBar.Left) - 120) + 50
    Parts(1) = ((rctClnt.Right - rctSBar.Left) - 120) + 100

    'パーツを2個に設定
    Res = Api_SendMessageByAny(SBhWnd, SB_SETPARTS, 2, Parts(0))
```

```
    'テキスト(パーツ0)
    Res = Api_SendMessageByString(SBhWnd, SB_SETTEXTA, 0, "CreateStatusWindow")

    'テキスト(パーツ1)
    Res = Api_SendMessageByString(SBhWnd, SB_SETTEXTA, 1, "TOKO")

    'ステータスバーのサイズを調整
    Res = Api_MoveWindow(SBhWnd, 0, rctClnt.Bottom - rctSBar.Bottom, 0, 0, True)
End Sub

'=======================================
'=
'=======================================
Declare Sub MainForm_QueryClose edecl (Cancel%)
Sub MainForm_QueryClose(cancel%)
    Var Res As Long

    If Cancel% = 0 Then

        'ステータスバーをアンロード
        Res = Api_DestroyWindow(SBhWnd)
    End If
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

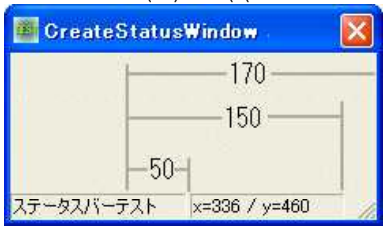
ステータスバーの作成(II)

タイマーを使用してマウス位置を表示させてみました。ステータスバーの表示・非表示、サイズグリップの表示非表示を切り替えています。

InitCommonControls コモンコントロールの初期化
CreateStatusWindow ステータスバーの作成
GetClientRectウィンドウのクライアント領域の座標を取得
MoveWindow ウィンドウの位置とサイズを変更
DestroyWindow ウィンドウの破棄
SendMessage (Api_SendMessageByAny)パーツの個数設定
SendMessage (Api_SendMessageByString)パーツにテキストを設定

矩形位置の計算

```
    Parts(0) = ((rctClnt.Right - rctSBar.Left) - 170) + 50
    Parts(1) = ((rctClnt.Right - rctSBar.Left) - 170) + 150
```



```
'====================================
'= ステータスバーを作成(II)
'=    タイマー使用でXYの位置を表示
'=    (CreateStatusWindow.bas)
'====================================
#include "Windows.bi"

Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

Type POINTAPI
    X   As Long
    Y   As Long
End Type

' マウスカーソル(マウスポインタ)の現在の位置に相当するスクリーン座標を取得
Declare Function Api_GetCursorPos& Lib "user32" Alias "GetCursorPos" (lpPoint As
POINTAPI)

' コモンコントロールライブラリからコモンコントロールのウィンドウクラスを登録して初期化
Declare Sub Api_InitCommonControls Lib "comctl32" Alias "InitCommonControls" ()

' ステータスバーを作成
Declare Function Api_CreateStatusWindow& Lib "Comctl32" Alias "CreateStatusWindowA"
(ByVal Style&, ByVal lpszText$, ByVal hWndParent&, ByVal wID&)

' ウィンドウのクライアント領域の座標を取得
Declare Function Api_GetClientRect& Lib "user32" Alias "GetClientRect" (ByVal hWnd&,
lpRect As RECT)

' 指定されたウィンドウの位置およびサイズを変更
Declare Function Api_MoveWindow& Lib "user32" Alias "MoveWindow" (ByVal hWnd&, ByVal X&,
ByVal Y&, ByVal nWidth&, ByVal nHeight&, ByVal bRepaint&)

' CreateWindowExの解放
Declare Function Api_DestroyWindow& Lib "user32" Alias "DestroyWindow" (ByVal hWnd&)

' ウィンドウにメッセージを送信
Declare Function Api_SendMessageByAny& Lib "user32" Alias "SendMessageA" (ByVal hWnd&,
ByVal wMsg&, ByVal wParam&, lParam As Any)

' パーツにテキストを設定
Declare Function Api_SendMessageByString& Lib "user32" Alias "SendMessageA" (ByVal
hWnd&, ByVal wMsg&, ByVal wParam&, ByVal lParam$)

#define WM_USER &H400                       'ユーザーが定義できるメッセージの使用領域を表す
#define WS_CHILD &H40000000                 '親ウインドウを持つコントロール(子ウインドウ)を作成
#define WS_VISIBLE &H10000000               '可視状態のウィンドウを作成する
#define WS_BORDER &H800000                  'フォームの枠線がある
#define SBARS_SIZEGRIP &H100                'サイズグリップ
#define CCS_BOTTOM &H3                      'ウィンドウを親ウィンドウの下端に配置
#define SB_SETTEXTA &H401                   'WM_USER + 1
#define SB_SETPARTS &H404                   'WM_USER + 4

Var Shared SBhWnd As Long
Var Shared Title As String

Var Shared Check1 As Object
Var Shared Check2 As Object
Var Shared Timer1 As Object

Check1.Attach GetDlgItem("Check1") : Check1.SetFontSize 14
Check2.Attach GetDlgItem("Check2") : Check2.SetFontSize 14
Timer1.Attach GetDlgItem("Timer1")
```

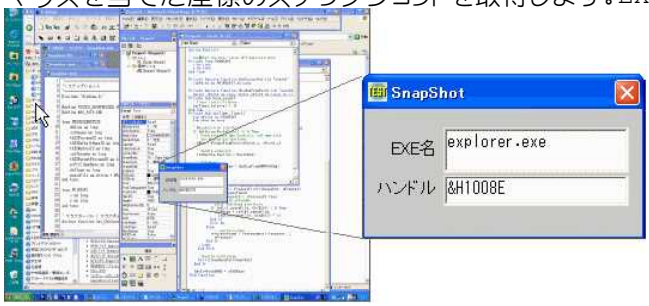
```
'=====================================
'=
'=====================================
Declare Sub DspOnOff()
Sub DspOnOff()
    Var rctSBar As RECT                         'ステータスバー矩形
    Var rctClnt As RECT                         'フォーム矩形
    Var Parts(1) As Long                        'ステータスバー部品数
    Var Ret As Long

    'ステータスバーをアンロード
    Ret = Api_DestroyWindow(SBhWnd)

    If Check1.GetCheck = 0 Then                 'ステータスバー非表示

        'サイズグリップ変更不許可
        Check2.EnableWindow 0
    Else If Check1.GetCheck = 1 Then            'ステータスバー表示

        If Check2.GetCheck = 0 Then

            SBhWnd = Api_CreateStatusWindow(WS_CHILD Or WS_VISIBLE Or WS_BORDER Or
CCS_BOTTOM, Title, GethWnd, 101)
        Else

            SBhWnd = Api_CreateStatusWindow(WS_CHILD Or WS_VISIBLE Or WS_BORDER Or
CCS_BOTTOM Or SBARS_SIZEGRIP, Title, GethWnd, 101)
        End If

        'サイズグリップ変更許可
        Check2.EnableWindow -1
    End If

    'メインフォームの矩形を取得
    Ret = Api_GetClientRect(GethWnd, rctClnt)

    'ステータスバーウィンドウの矩形を取得
    Ret = Api_GetClientRect(SBhWnd, rctSBar)

    Parts(0) = ((rctClnt.Right - rctSBar.Left) - 170) + 50
    Parts(1) = ((rctClnt.Right - rctSBar.Left) - 170) + 150

    'パーツを2個に設定
    Ret = Api_SendMessageByAny(SBhWnd, SB_SETPARTS, 2, Parts(0))

    'ステータスバーのサイズを調整
    Ret = Api_MoveWindow(SBhWnd, 0, rctClnt.Bottom - rctSBar.Bottom, 0, 0, True)
End Sub

'=====================================
'=
'=====================================
Declare Sub Mainform_Start edecl ()
Sub Mainform_Start()
    Var rc As RECT
    Var cBrush As Long
    Var rgbColor As Long
    Var Ret As Long

    '表面色を設定
    rgbColor = RGB(236, 233, 216)

    'Mainformを取得色で塗り
    SetBackColor rgbColor

    'Mainformを表示
    ShowWindow -1

    '画面を消去
    Cls
```

```
    Api_InitCommonControls

    Timer1.SetInterval 10
    Timer1.Enable -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Mainform_Resize edecl ()
Sub Mainform_Resize()
    DspOnOff
End Sub

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var lpPoint As POINTAPI
    Var Ret As Long

    Ret = Api_GetCursorPos(lpPoint)

    'テキスト
    Ret = Api_SendMessageByString(SBhWnd, SB_SETTEXTA, 0, "ステータスバーテスト")
    Ret = Api_SendMessageByString(SBhWnd, SB_SETTEXTA, 1, "x=" & Trim$(Str$(lpPoint.x)) &
" / y=" & Trim$(Str$(lpPoint.y)))
End Sub

'=====================================
'=
'=====================================
Declare Sub Check1_ON edecl ()
Sub Check1_ON()
    DspOnOff
End Sub

'=====================================
'=
'=====================================
Declare Sub Check2_ON edecl ()
Sub Check2_ON()
    DspOnOff
End Sub

'=====================================
'=
'=====================================
Declare Sub Mainform_QueryClose edecl (Cancel%)
Sub Mainform_QueryClose(cancel%)
    Var Ret As Long
    If Cancel% = 0 Then

        'ステータスバーをアンロード
        Ret = Api_DestroyWindow(SBhWnd)
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

スナップショットを取得(Ⅰ)

GetCursorPos マウスカーソル(マウスポインタ)の現在の位置に相当するスクリーン座標を取得
WindowFromPoint 指定の座標位置にあるウィンドウハンドルを取得
GetWindowThreadProcessId ウィンドウのプロセスIDとスレッドIDを取得
CreateToolhelp32Snapshot プロセスのスナップショットを取得
Process32First 最初のプロセスに関する情報を取得
Process32Next 2番目以降のプロセスに関する情報を取得
CloseHandle オープンされているオブジェクトハンドルをクローズ

マウスを当てた座標のスナップショットを取得します。EXE名とそのハンドルを表示します。




```
'=======================================
'= スナップショット
'=    (SnapShot.bas)
'=======================================
#include "Windows.bi"

#define TH32CS_SNAPPROCESS &H2              'プロセス一覧のスナップショット
#define MAX_PATH 260

Type PROCESSENTRY32
    dwSize                  As Long
    cntUsage                As Long
    th32ProcessID           As Long
    th32DefaultHeapID       As Long
    th32ModuleID            As Long
    cntThreads              As Long
    th32ParentProcessID     As Long
    pcPriClassBase          As Long
    dwflags                 As Long
    szexeFile               As String * MAX_PATH
End Type

Type POINTAPI
    x   As Long
    y   As Long
End Type

' マウスカーソル(マウスポインタ)の現在の位置に相当するスクリーン座標を取得
Declare Function Api_GetCursorPos& Lib "user32" Alias "GetCursorPos" (lpPoint As
POINTAPI)

' 指定の座標位置にあるウィンドウハンドルを取得
Declare Function Api_WindowFromPoint& Lib "user32" Alias "WindowFromPoint" (ByVal
xPoint&, ByVal yPoint&)

' ウィンドウのプロセスIDとスレッドIDを取得
Declare Function Api_GetWindowThreadProcessId& Lib "user32" Alias
"GetWindowThreadProcessId" (ByVal hWnd&, lpdwProcessId&)

' プロセスのスナップショットを取得
Declare Function Api_CreateToolhelp32Snapshot& Lib "kernel32" Alias
"CreateToolhelp32Snapshot" (ByVal dwFlag&, ByVal th32ProcessID&)

' 最初のプロセスに関する情報を取得する関数
Declare Function Api_Process32First& Lib "kernel32" Alias "Process32First" (ByVal
hSnapshot&, lppe As PROCESSENTRY32)
```

```
' 2番目以降のプロセスに関する情報を取得する関数
Declare Function Api_Process32Next& Lib "kernel32" Alias "Process32Next" (ByVal
hSnapshot&, lppe As PROCESSENTRY32)

' オープンされているオブジェクトハンドルをクローズ
Declare Function Api_CloseHandle& Lib "kernel32" Alias "CloseHandle" (ByVal hObject&)

Var Shared Text(1) As Object
Var Shared Edit(1) As Object
Var Shared Timer1 As Object

For i = 0 To 1
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1))) : Text(i).SetFontSize 14
    Edit(i).Attach GetDlgItem("Edit" & Trim$(Str$(i + 1))) : Edit(i).SetFontSize 14
Next
Timer1.Attach GetDlgItem("Timer1")

'=====================================
'=
'=====================================
Declare Function GetExeFromHWND(hWnd As Long) As String
Function GetExeFromHWND(hWnd As Long) As String
    Var ThreadID As Long
    Var ProcessID As Long
    Var Snapshot As Long
    Var Process As PROCESSENTRY32
    Var ProcessFound As Long
    Var ExeName As String
    Var Ret As Long

    'プロセスID取得
    ThreadID = Api_GetWindowThreadProcessId(hWnd, ProcessID)

    If ThreadID <> 0 and ProcessID <> 0 Then

        'スナップショット取得
        Snapshot = Api_CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0)

        If Snapshot = -1 Then Exit Function

        'プロセスサイズ
        Process.dwSize = Len(Process)

        '最初に見つかった
        ProcessFound = Api_Process32First(Snapshot, Process)

        Do While ProcessFound
            If Process.th32ProcessID = ProcessID Then

                'プロセスIDが見つかった場合、そのEXE名を取得
                If InStr(Process.szexeFile, Chr$(0)) > 0 Then
                    ExeName = Left$(Process.szexeFile, InStr(Process.szexeFile, Chr$(0)) -
1)
                End If
                Exit Do
            Else

                '見つからない場合次を探す
                ProcessFound = Api_Process32Next(Snapshot, Process)
            End If
        Loop

        'ハンドルクローズ
        Ret = Api_CloseHandle(Snapshot)
    End If

    GetExeFromHWND = ExeName
End Function
```

```
'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Timer1.SetInterval 10
    Timer1.Enable -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var pa As POINTAPI
    Var hWnd As Long

    'マウス位置取得
    If Api_GetCursorPos(pa) <> 0 Then

        'マウスカーソル座標のハンドル取得
        hWnd = Api_WindowFromPoint(pa.x, pa.y)

        'EXE名
        Edit(0).SetWindowText GetExeFromHWND(hWnd)

        'ハンドル
        Edit(1).SetWindowText "&H" & hex$(hWnd)
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

スナップショットを取得(II)

SendInput キーストローク、マウスの動き、ボタンのクリックなどを合成
OpenClipboard クリップボードをオープン
IsClipboardFormatAvailable 指定したフォーマットがクリップボードにあるかどうか判定
GetClipboardData クリップボードから指定フォーマットのデータを検索
CloseClipboard クリップボードをクローズ
CreateCompatibleDC メモリデバイスコンテキストを作成
SelectObject 指定されたデバイスコンテキストのオブジェクトを選択
GetObject オブジェクト取得
BitBlt ビットブロック転送
DeleteDC 指定されたデバイスコンテキストを削除
GetDC デバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放

「PrintScreen」キーの操作に相当するイベントを擬似的に発生させ、クリップボードに画面全体のスナップショットを取得。
「表示」で、PictureBoxに表示させています。

```
'====================================
'= スナップショットを取得(II)
'=    (SendInput.bas)
'====================================
#include "Windows.bi"

Type KEYBDINPUT
    wVk                 As Integer
    wScan               As Integer
    dwFlags             As Long
    ntime               As Long
    dwExtraInfo         As Long
    bytUnusedPadding(7) As Byte
End Type

Type tagINPUT
    dwtype     As Long
    ki         As KEYBDINPUT
End Type

Type BITMAP
    bmType              As Long
    bmWidth             As Long
    bmHeight            As Long
    bmWidthBytes        As Long
    bmPlanes            As Integer
    bmBitsPixel         As Integer
    bmBits              As Long
End Type

#define INPUT_KEYBOARD 1
#define INPUT_HARDWARE 2
#define VK_SNAPSHOT &H2C                    '[Snap Shot]
#define KEYEVENTF_KEYUP &H2                 'キーを放す
#define CF_BITMAP 2                         'ビットマップのデータ(HBITMAP)
#define SRCCOPY &HCC0020                    'そのまま転送

' キーストローク、マウスの動き、ボタンのクリックなどを合成
Declare Function Api_SendInput& Lib "user32" Alias "SendInput" (ByVal cInputs&, ByRef
pInputs As tagINPUT, ByVal cbSize&)

' クリップボードをオープン
Declare Function Api_OpenClipboard& Lib "user32" Alias "OpenClipboard" (ByVal hWnd&)

' 指定したフォーマットがクリップボードにあるかどうか判定
Declare Function Api_IsClipboardFormatAvailable& Lib "user32" Alias
"IsClipboardFormatAvailable" (ByVal wFormat&)

' クリップボードから指定フォーマットのデータを検索
Declare Function Api_GetClipboardData& Lib "user32" Alias "GetClipboardData" (ByVal
wFormat&)

' クリップボードをクローズ
Declare Function Api_CloseClipboard& Lib "user32" Alias "CloseClipboard" ()

' 指定されたデバイスコンテキストに関連するデバイスと互換性のあるメモリデバイスコンテキストを作成
Declare Function Api_CreateCompatibleDC& Lib "gdi32" Alias "CreateCompatibleDC" (ByVal
hDC&)

' 指定されたデバイスコンテキストのオブジェクトを選択
Declare Function Api_SelectObject& Lib "gdi32" Alias "SelectObject" (ByVal hDC&, ByVal
hObject&)

' オブジェクト取得
Declare Function Api_GetObject& Lib "gdi32" Alias "GetObjectA" (ByVal hObject&, ByVal
nCount&, lpObject As Any)

' ビットブロック転送を行う。コピー元からコピー先のデバイスコンテキストへ、指定された長方形内の各ピクセルの色
データをコピー
Declare Function Api_BitBlt& Lib "gdi32" Alias "BitBlt" (ByVal hDestDC&, ByVal X&, ByVal
```

```
Y&, ByVal nWidth&, ByVal nHeight&, ByVal hSrcDC&, ByVal xSrc&, ByVal ySrc&, ByVal dwRop&)

' 指定されたデバイスコンテキストを削除
Declare Function Api_DeleteDC& Lib "gdi32" Alias "DeleteDC" (ByVal hDC&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

Var Shared Text1 As Object
Var Shared Picture1 As Object
Var Shared Button1 As Object
Var Shared Button2 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Picture1.Attach GetDlgItem("Picture1") : Picture1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var ta(1) As tagINPUT
    Var Ret As Long

    'インプットイベントの種類を指定
    ta(0).dwtype = INPUT_KEYBOARD

    '仮想キーコードを指定
    ta(0).ki.wVk = VK_SNAPSHOT

    'インプットイベントの種類を指定
    ta(1).dwtype = INPUT_KEYBOARD

    '仮想キーコードを指定
    ta(1).ki.wVk = VK_SNAPSHOT

    '動作を指定
    ta(1).ki.dwFlags = KEYEVENTF_KEYUP

    'キーストロークを合成
    Ret = Api_SendInput(2, ta(0), Len(ta(0)))
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var bmp As BITMAP
    Var hBit As Long
    Var phDC As Long
    Var mhDC As Long
    Var Ret As Long

    phDC = Api_GetDC(Picture1.GethWnd)

    Picture1.Cls

    'Bitmap型式データの有無を調査
    If Api_IsClipboardFormatAvailable(CF_BITMAP) <> 0 Then
        Ret = Api_OpenClipboard(GethWnd)

        '指定フォーマットのBITMAPデータを検索
        hBit = Api_GetClipboardData(CF_BITMAP)
```

```
        'メモリデバイスコンテキストを作成
        mhDC = Api_CreateCompatibleDC(phDC)

        'Object取得
        Ret = Api_GetObject(hBit, Len(bmp), bmp)

        'Object選択
        Ret = Api_SelectObject(mhDC, hBit)

        '指定の(PictureBox)のデバイスコンテキストにメモリデバイスコンテキストのデータを転送
        Ret = Api_BitBlt(phDC, 0, 0, bmp.bmWidth, bmp.bmHeight, mhDC, 0, 0, SRCCOPY)

        Ret = Api_ReleaseDC(Picture1.GethWnd, phDC)
        Ret = Api_DeleteDC(mhDC)
        Ret = Api_CloseClipboard()
    End If
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

スプライト処理テスト(Ⅰ)

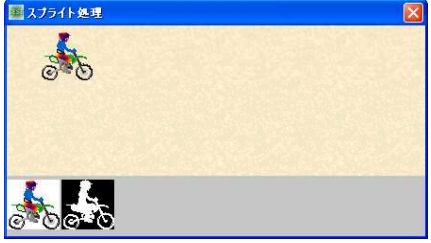
にわか勉強のスプライト処理テストであることをお断りしておきます。
BitBlt 画像転送
GetDC デバイスコンテキスト取得
ReleaseDC デバイスコンテキスト解放

Form設定

PictureBoxを3個とTimerを貼り付けます。
Picture1(450×160ピクセル)backcolor.bmp:背景色用(450×80ピクセル)
実は、Picture1に下記背景色を上下2段に読み込んでいます。(bikeを表示した後、次を表示させるために一度消去しなければならないのですが、背景色の模様を合わせるため)

Picture2(56×52ピクセル):bike.bmp:バイク移動用
Picture3(56×52ピクセル):bike2.bmp:バイクマスク用


まだまだ未完成で、タイマーインターバル、移動量等を変えてテストしていますがフリッカが出るようです。



参照
スプライト処理②

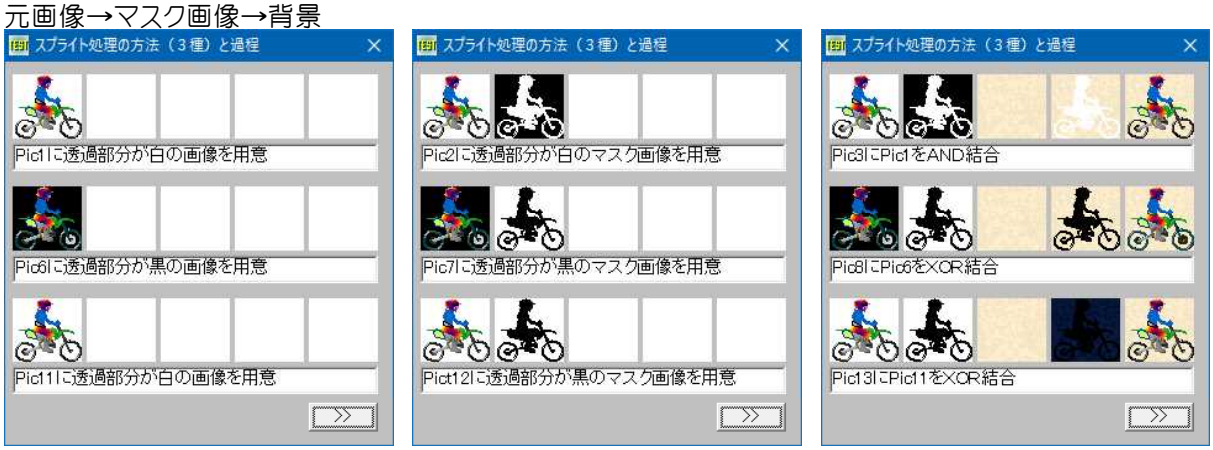
```
'====================================
'= スプライト処理テスト
'=    (Sprite.bas)
'====================================
#include "Windows.bi"

' ビットブロック転送を行う。コピー元からコピー先のデバイスコンテキストへ、指定された長方形内の各ピクセルの色データをコピー
Declare Function Api_BitBlt& Lib "gdi32" Alias "BitBlt" (ByVal hDestDC&, ByVal X&, ByVal Y&, ByVal nWidth&, ByVal nHeight&, ByVal hSrcDC&, ByVal xSrc&, ByVal ySrc&, ByVal dwRop&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

#define MERGEPAINT &HBB0226                 '反転した転送元ビットマップとパターンビットマップを論理OR演算子で結合
#define SRCAND &H8800C6                     '転送先の画像とAND演算して転送
#define SRCCOPY &HCC0020                    'そのまま転送
#define SRCERASE &H440328                   '転送先ビットマップを反転、その結果と転送元ビットマップを論理AND演算子で結合
#define SRCINVERT &H660046                  '転送先の画像とXOR演算して転送
#define SRCPAINT &HEE0086                   '転送先の画像とOR演算して転送

Var shared Picture1 As Object : Picture1.Attach GetDlgItem("Picture1")
Var shared Picture2 As Object : Picture2.Attach GetDlgItem("Picture2")
Var shared Picture3 As Object : Picture3.Attach GetDlgItem("Picture3")
Var shared Timer1 As Object : Timer1.Attach GetDlgItem("Timer1")
Var shared Bitmap As Object : BitmapObject Bitmap

Var shared hDC1 As Long
Var shared hDC2 As Long
Var shared hDC3 As Long
Var shared X As Integer
Var shared Y As Integer
Var shared W As Long
Var shared H As Long
Var shared M As Long

'====================================
'=
'====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var Ret As Long

    X = 0                                   '最初のX位置
    Y = 15                                  '最初のY位置
    M = 5                                   '移動量
    W = Picture2.GetWidth
    H = Picture2.GetHeight

    'Picture1上半分にbackcolor.bmpを読み込む
    Bitmap.LoadFile "backcolor.bmp"
    Picture1.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject

    'Picture1下半分にbackcolor.bmpを読み込む(苦し紛れのbike消去用)
    Bitmap.LoadFile "backcolor.bmp"
    Picture1.DrawBitmap Bitmap, 0, 80
    Bitmap.DeleteObject

    'Picture2にbike1.bmpを読み込む
    Bitmap.LoadFile "bike1.bmp"
    Picture2.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject
```

```
    'Picture3にbike2.bmp(白抜き)を読み込む
    Bitmap.LoadFile "bike2.bmp"
    Picture3.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject

    'Picture毎のDC取得
    hDC1 = Api_GetDC(Picture1.GethWnd)
    hDC2 = Api_GetDC(Picture2.GethWnd)
    hDC3 = Api_GetDC(Picture3.GethWnd)

    Timer1.SetInterval 10
    Timer1.Enable -1
end sub

'====================================
'= 移動処理
'====================================
Declare Sub Bike_Move edecl ()
Sub Bike_Move()
    Var Ret As Long

    'Picture1の下半分の同位置画像を上半分にコピー(無理やりbike消去^^;)
    Ret = Api_BitBlt(hDC1, X - M, 0, W, 80, hDC1, X - M, 80, SRCCOPY)

    'Picture1の画像とPicture3のマスク画像(bike2.bmp)をORで合成(SRCPAINT)=白抜き画像
    Ret = Api_BitBlt(hDC1, X, Y, W, H, hDC3, 0, 0, SRCPAINT)

    'Picture1の画像とPicture2の画像(bike1.bmp)をANDで合成(SRCAND)
    Ret = Api_BitBlt(hDC1, X, Y, W, H, hDC2, 0, 0, SRCAND)
End Sub

'====================================
'=
'====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    X = X + M : If X > GetWidth + M Then X = 0
    Y = Int(Rnd(1) * 5) + 5
    Bike_Move
End Sub

'====================================
'=
'====================================
Declare Sub MainForm_QueryClose(Cancel As Integer, ByVal Mode As Integer)
Sub MainForm_QueryClose(Cancel As Integer, ByVal Mode As Integer)
    Var Ret As Long

    If Cancel = 0 Then
        Ret = Api_ReleaseDC(Picture1.GethWnd, hDC1)
        Ret = Api_ReleaseDC(Picture2.GethWnd, hDC2)
        Ret = Api_ReleaseDC(Picture3.GethWnd, hDC3)
        End
    End If
End Sub

'====================================
'=
'====================================
While 1
    WaitEvent
Wend
Stop
End
```

スプライト処理テスト（II）3通りの方法

元画像の透過処理部、およびマスク画像の透過処理部をそれぞれ黒または白に指定した場合の経過をたどってみます。
BitBlt 画像転送
GetDC デバイスコンテキスト取得
ReleaseDC デバイスコンテキスト解放

元画像→マスク画像→背景




```
'=====================================
'= スプライト処理テスト(3種の方法と経過の把握)
'=    (Sprite4.bas)
'=====================================
#include "Windows.bi"

' ビットブロック転送を行う。コピー元からコピー先のデバイスコンテキストへ、指定された長方形内の各ピクセルの色データをコピー
Declare Function Api_BitBlt& Lib "gdi32" Alias "BitBlt" (ByVal hDestDC&, ByVal X&, ByVal Y&, ByVal nWidth&, ByVal nHeight&, ByVal hSrcDC&, ByVal xSrc&, ByVal ySrc&, ByVal dwRop&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

#define SRCCOPY &HCC0020                'コピー元をコピー先にそのままコピー
#define SRCPAINT &HEE0086               'コピー元とコピー先を論理OR演算子で結合
#define SRCAND &H8800C6                 'コピー元とコピー先を論理AND演算子で結合
#define SRCINVERT &H660046              'コピー元とコピー先を論理XOR演算子で結合
#define SRCERASE &H440328               'コピー先の色を反転した色とコピー元の色を論理AND演算子で結合

Var shared Picture(14) As Object
Var shared Text(2) As Object
Var shared Button1 As Object
Var shared Bitmap As Object
BitmapObject Bitmap

For i = 0 To 14
    Picture(i).Attach GetDlgItem("Picture" & Trim$(Str$(i + 1)))
Next
For i = 0 To 2
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1)))
    Text(i).SetFontSize 14
Next
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var shared hDC(14) As Long
Var shared W As Long
Var shared H As Long
Var shared Job As byte
```

```
'====================================
'=
'====================================
Declare Sub Job1 edecl ()
Sub Job1()
    Bitmap.LoadFile "bike1.bmp"
    Picture(0).DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject
    Text(0).SetWindowText "Pic1に透過部分が白の画像を用意"

    Bitmap.LoadFile "bike4.bmp"
    Picture(5).DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject
    Text(1).SetWindowText "Pic6に透過部分が黒の画像を用意"

    Bitmap.LoadFile "bike1.bmp"
    Picture(10).DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject
    Text(2).SetWindowText "Pic11に透過部分が白の画像を用意"
End Sub

'====================================
'=
'====================================
Declare Sub Job2 edecl ()
Sub Job2()
    Bitmap.LoadFile "bike2.bmp"
    Picture(1).DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject
    Text(0).SetWindowText "Pic2に透過部分が白のマスク画像を用意"

    Bitmap.LoadFile "bike3.bmp"
    Picture(6).DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject
    Text(1).SetWindowText "Pic7に透過部分が黒のマスク画像を用意"

    Bitmap.LoadFile "bike3.bmp"
    Picture(11).DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject
    Text(2).SetWindowText "Pict12に透過部分が黒のマスク画像を用意"
End Sub

'====================================
'=
'====================================
Declare Sub Job3 edecl ()
Sub Job3()
    Bitmap.LoadFile "backcol.bmp"
    Picture(2).DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject
    Text(0).SetWindowText "Pic3に背景画像を用意"

    Bitmap.LoadFile "backcol.bmp"
    Picture(7).DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject
    Text(1).SetWindowText "Pic8に背景画像を用意"

    Bitmap.LoadFile "backcol.bmp"
    Picture(12).DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject
    Text(2).SetWindowText "Pic13に背景画像を用意"
End Sub

'====================================
'=
'====================================
Declare Sub Job4 edecl ()
Sub Job4()
    Ret = Api_BitBlt(hDC(3), 0, 0, W, H, hDC(2), 0, 0, SRCCOPY)
    Ret = Api_BitBlt(hDC(3), 0, 0, W, H, hDC(1), 0, 0, SRCPAINT)
```

```
    Text(0).SetWindowText "Pic3にPic2をOR結合"

    Ret = Api_BitBlt(hDC(8), 0, 0, W, H, hDC(7), 0, 0, SRCCOPY)
    Ret = Api_BitBlt(hDC(8), 0, 0, W, H, hDC(6), 0, 0, SRCAND)
    Text(1).SetWindowText "Pic8にPic7をAND結合"

    Ret = Api_BitBlt(hDC(13), 0, 0, W, H, hDC(12), 0, 0, SRCCOPY)
    Ret = Api_BitBlt(hDC(13), 0, 0, W, H, hDC(11), 0, 0, SRCERASE)
    Text(2).SetWindowText "Pic13に反転したPic12をAND結合"
End Sub

'=====================================
'=
'=====================================
Declare Sub Job5 edecl ()
Sub Job5()
    Ret = Api_BitBlt(hDC(4), 0, 0, W, H, hDC(2), 0, 0, SRCCOPY)
    Ret = Api_BitBlt(hDC(4), 0, 0, W, H, hDC(1), 0, 0, SRCPAINT)
    Ret = Api_BitBlt(hDC(4), 0, 0, W, H, hDC(0), 0, 0, SRCAND)
    Text(0).SetWindowText "Pic3にPic1をAND結合"

    Ret = Api_BitBlt(hDC(9), 0, 0, W, H, hDC(7), 0, 0, SRCCOPY)
    Ret = Api_BitBlt(hDC(9), 0, 0, W, H, hDC(6), 0, 0, SRCAND)
    Ret = Api_BitBlt(hDC(9), 0, 0, W, H, hDC(5), 0, 0, SRCINVERT)
    Text(1).SetWindowText "Pic8にPic6をXOR結合"

    Ret = Api_BitBlt(hDC(14), 0, 0, W, H, hDC(12), 0, 0, SRCCOPY)
    Ret = Api_BitBlt(hDC(14), 0, 0, W, H, hDC(11), 0, 0, SRCERASE)
    Ret = Api_BitBlt(hDC(14), 0, 0, W, H, hDC(10), 0, 0, SRCINVERT)
    Text(2).SetWindowText "Pic13にPic11をXOR結合"
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Job = 0

    W = Picture(1).GetWidth
    H = Picture(1).GetHeight

    For i = 0 To 14
        hDC(i) = Api_GetDC(Picture(i).GethWnd)
    Next
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_ON edecl ()
Sub Button1_ON()
    Job = Job + 1
    Select Case Job
        Case 1
            Job1
        Case 2
            Job2
        Case 3
            Job3
        Case 4
            Job4
        Case 5
            Job5
        Case 6
            For i = 0 To 14
                Picture(i).Cls
            Next
            For i = 0 To 2
                Text(i).SetWindowText ""
```

```
            Next
            Job = 0
    End Select
End Sub

'=======================================
'=
'=======================================
Declare Sub MainForm_QueryClose(Cancel%, ByVal Mode%)
Sub MainForm_QueryClose(Cancel%, ByVal Mode%)
    Var Ret As Long

    If Cancel% = 0 Then
        For i = 0 To 14
            Ret = Api_ReleaseDC(Picture(i).GethWnd, hDC(i))
        Next
        End
    End If
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

スレッドIDとプロセスIDを取得

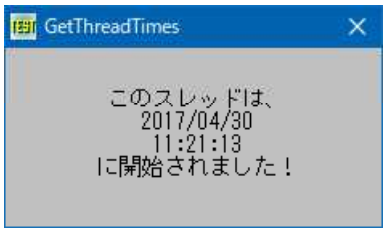
自フォームを作成したスレッドIDとプロセスIDを取得します。
GetWindowThreadProcessId ウィンドウのプロセスIDとスレッドIDを取得


```
'=======================================
'= スレッドID・プロセスIDを取得
'=    (GetWindowThreadProcessId.bas)
'=======================================
#include "Windows.bi"

' ウィンドウのプロセスIDとスレッドIDを取得する関数の宣言
Declare Function Api_GetWindowThreadProcessId& Lib "user32" Alias
"GetWindowThreadProcessId" (ByVal hWnd&, lpdwProcessId&)

Var Shared Text(3) As Object

For i = 0 To 3
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1)))
    Text(i).SetFontSize 14
Next

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var ThreadId As Long
    Var ProcessId As Long
```

```
    'スレッドIDとプロセスIDを取得
    ThreadId = Api_GetWindowThreadProcessId(GethWnd, ProcessId)

    'スレッドIDを表示
    Text(2).SetWindowText "&&H" & Hex$(ThreadId)

    'プロセスIDを表示
    Text(3).SetWindowText "&&H" & Hex$(ProcessId)
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

スレッドの開始時間を取得

GetThreadTimes 指定されたスレッドのタイミングを取得
FileTimeToLocalFileTime 世界協定時刻形式ファイル時間をローカルファイル時間に変換
FileTimeToSystemTime ファイルタイムをシステムタイムに変換
GetCurrentProcess 現在のプロセスに対応する疑似ハンドルを取得




```
'=======================================
'= スレッドの開始時間を取得
'= WindowsNT3.1以降
'=======================================
#include "Windows.bi"

Type FILETIME
    dwLowDateTime     As Long
    dwHighDateTime    As Long
End Type

Type SYSTEMTIME
    wYear             As Integer
    wMonth            As Integer
    wDayOfWeek        As Integer
    wDay              As Integer
    wHour             As Integer
    wMinute           As Integer
    wSecond           As Integer
    wMilliseconds     As Integer
End Type

' 指定されたスレッドのタイミング情報を取得
Declare Function Api_GetThreadTimes& Lib "kernel32" Alias "GetThreadTimes" (ByVal
hThread&, lpCreationTime As FILETIME, lpExitTime As FILETIME, lpKernelTime As FILETIME,
lpUserTime As FILETIME)

' 世界協定時刻形式ファイル時間をローカルファイル時間に変換
Declare Function Api_FileTimeToLocalFileTime& Lib "kernel32" Alias
"FileTimeToLocalFileTime" (lpFileTime As FILETIME, lpLocalFileTime As FILETIME)
```

```
' ファイルタイムをシステムファイルに変換
Declare Function Api_FileTimeToSystemTime& Lib "kernel32" Alias "FileTimeToSystemTime"
(lpFileTime As FILETIME, lpSystemTime As SYSTEMTIME)

' カレントスレッドの擬似ハンドルを取得
Declare Function Api_GetCurrentThread& Lib "kernel32" Alias "GetCurrentThread" ()

Var Shared Text1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14

'=======================================
'=
'=======================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var FT0 As FILETIME
    Var FT1 As FILETIME
    Var ST As SYSTEMTIME
    Var txt As String
    Var Ret As Long

    Ret = Api_GetThreadTimes(Api_GetCurrentThread, FT1, FT0, FT0, FT0)
    Ret = Api_FileTimeToLocalFileTime(FT1, FT1)
    Ret = Api_FileTimeToSystemTime(FT1, ST)

    txt = txt & "このスレッドは、" & Chr$(13, 10)
    txt = txt & Trim$(Str$(ST.wYear)) & "/" & Right$(Str$(100 + ST.wMonth), 2) & "/" &
Right$(Str$(100 + ST.wDay), 2) & Chr$(13, 10)
    txt = txt & Right$(Str$(100 + ST.wHour), 2) & ":" & Right$(Str$(100 + ST.wMinute), 2) &
":" & Right$(Str$(100 + ST.wSecond), 2) & Chr$(13, 10)
    txt = txt & "に開始されました！"
    Text1.SetWindowText txt
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

スレッドの情報を列挙

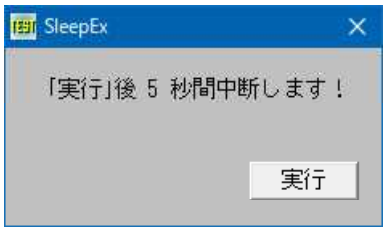
スレッドの情報を列挙します。
CreateToolhelp32Snapshot プロセスのスナップショットを取得
CloseHandle オブジェクトハンドルをクローズ
GetCurrentProcessId 自分自身のプロセスIDを取得
Thread32First 最初のスレッド情報を取得
Thread32Next 次のスレッド情報を取得

```
'=====================================
'= スレッドの情報の列挙
'=    (Thread32First.bas)
'=====================================
#include "Windows.bi"

#define TH32CS_INHERIT -2147483648          '返されるハンドルを継承可能とする
#define TH32CS_SNAPALL &HF                  'TH32CS_SNAPHEAPLIST･SNAPMODULE･SNAPPROCESS･
                                            SNAPTHREADの組み合わせ
#define TH32CS_SNAPHEAPLIST &H1             'ProcessIDで指定したプロセスのヒープリスト
#define TH32CS_SNAPMODULE &H8               'ProcessIDで指定したプロセスのモジュール一覧のスナッ
                                            プショット
#define TH32CS_SNAPPROCESS &H2              'プロセス一覧のスナップショット
#define TH32CS_SNAPTHREAD &H4               'スレッド一覧のスナップショット

Type THREADENTRY32
    dwSize            As Long               '構造体サイズ
    cntUsage          As Long               '参照カウント(0)
    th32ThreadID      As Long               'スレッドID
    th32OwnerProcessID As Long              'プロセスID
    tpBasePri         As Long               '基本優先順位レベル
    tpDeltaPri        As Long               '(0)
    dwFlags           As Long               '(0)
End Type

' プロセスのスナップショットを取得
Declare Function Api_CreateToolhelp32Snapshot& Lib "Kernel32" Alias
"CreateToolhelp32Snapshot" (ByVal dwFlag&, ByVal th32ProcessID&)

' オープンされているオブジェクトハンドルをクローズ
Declare Function Api_CloseHandle& Lib "Kernel32" Alias "CloseHandle" (ByVal hObject&)

' 自分自身のプロセスIDを取得
Declare Function Api_GetCurrentProcessId& Lib "Kernel32" Alias "GetCurrentProcessId" ()

' 最初のスレッドの情報を取得
Declare Function Api_Thread32First& Lib "kernel32" Alias "Thread32First" (ByVal
hSnapshot&, lppe As THREADENTRY32)

' 次のスレッドの情報を取得
Declare Function Api_Thread32Next& Lib "kernel32" Alias "Thread32Next" (ByVal
hSnapshot&, lppe As THREADENTRY32)

Var Shared List1 As Object
Var Shared Button1 As Object

List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'= スレッドと優先順位
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var te As THREADENTRY32
    Var hSnap As Long
    Var pid As Long
    Var Ret As Long

    List1.Resetcontent

    'プロセスID取得
    pid = Api_GetCurrentProcessId()

    'プロセスのスナップショットを取得
    hSnap = Api_CreateToolhelp32Snapshot(TH32CS_SNAPTHREAD, pid)

    '構造体のサイズ
    te.dwSize = Len(te)
```

```
    '最初のスレッドの情報を取得
    Ret = Api_Thread32First(hSnap, te)
    If Ret <= 0 Then Exit Sub

    '全てのスレッドの情報を表示
    Do While Ret
        List1.AddString "ThreadID:" & Hex$(te.th32ThreadID) & " / Priority:" & Trim$(Str$
(te.tpBasePri))
        Ret = Api_Thread32Next(hSnap, te)
    Loop

    'オブジェクトハンドルをクローズ
    Ret = Api_CloseHandle(hSnap)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

スレッドを中断

SleepEx 現在のスレッドを中断




```
'=====================================
'= スレッドを中断
'=    (SleepEx.bas)
'=====================================
#include "Windows.bi"

' 現在のスレッドを中断
Declare Function Api_SleepEx& Lib "kernel32" Alias "SleepEx" (ByVal dwMilliseconds&,
ByVal bAlertable&)

Var Shared Text1 As Object
Var Shared Text2 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Text2.Attach GetDlgItem("Text2") : Text2.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Text1.SetWindowText "「実行」後 5 秒間中断します！"
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
```

```
Sub Button1_on()
    Var Ret As Long

    Text1.SetWindowtext "SleepEx 開始時間 " & Time$

    Ret = Api_SleepEx(5000, 1)

    Text2.SetWindowtext "SleepEx 終了時間 " & Time$
End Sub

'========================================
'=
'========================================
While 1
    WaitEvent
Wend
Stop
End
```

選択されたデバイスコンテキストのパスをリージョンに変換

CreateFontIndirect 論理フォントを作成
SelectObject 指定されたデバイスコンテキストのオブジェクトを選択
BeginPath hDCで指定されたデバイスコンテキストのパスを作成
EndPath BeginPathで開始したパスの作成を終了
PathToRegion hDCで指定したデバイスコンテキストで選択されているパスをリージョンに変換
TextOut 文字列を描画
SetWindowRgn 指定の領域をウィンドウ領域として設定
CreateSolidBrush ソリッドカラーで論理ブラシを作成
DeleteObject オブジェクトを削除
ReleaseCapture マウスキャプチャを解放
SendMessage ウィンドウにメッセージを送信
GetDC デバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放


```
'========================================
'= 選択されたDCパスをリージョンに変換
'=    (PathToRegion.bas)
'========================================
#include "Windows.bi"

#define LF_FACESIZE 32

Type LOGFONT
    lfHeight          As Long
    lfWidth           As Long
    lfEscapement      As Long
    lfOrientation     As Long
    lfWeight          As Long
    lfItalic          As Byte
    lfUnderline       As Byte
    lfStrikeOut       As Byte
    lfCharSet         As Byte
    lfOutPrecision    As Byte
    lfClipPrecision   As Byte
    lfQuality         As Byte
    lfPitchAndFamily  As Byte
    lfFaceName(LF_FACESIZE) As Byte
End Type
```

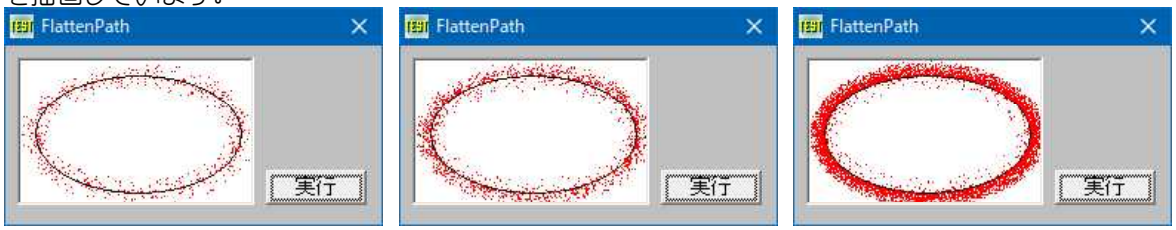
```
' 論理フォントを作成
Declare Function Api_CreateFontIndirect& Lib "gdi32" Alias "CreateFontIndirectA"
(lpLogFont As LOGFONT)

' 指定されたデバイスコンテキストのオブジェクトを選択
Declare Function Api_SelectObject& Lib "gdi32" Alias "SelectObject" (ByVal hDC&, ByVal
hObject&)

' hDCで指定されたデバイスコンテキストのパスの作成
Declare Function Api_BeginPath& Lib "gdi32" Alias "BeginPath" (ByVal hDC&)

' BeginPathで開始したパスの作成を終了
Declare Function Api_EndPath& Lib "gdi32" Alias "EndPath" (ByVal hDC&)

' hDCで指定したデバイスコンテキストで選択されているパスをリージョンに変換する
Declare Function Api_PathToRegion& Lib "gdi32" Alias "PathToRegion" (ByVal hDC&)

' 文字を描画
Declare Function Api_TextOut& Lib "gdi32" Alias "TextOutA" (ByVal hDC&, ByVal nXStart&,
ByVal nYStart&, ByVal lpString$, ByVal cbString&)

' 指定の領域をウィンドウ領域として設定
Declare Function Api_SetWindowRgn& Lib "user32" Alias "SetWindowRgn" (ByVal hWnd&, ByVal
hRgn&, ByVal bRedraw&)

' ソリッドカラーで論理ブラシを作成
Declare Function Api_CreateSolidBrush& Lib "gdi32" Alias "CreateSolidBrush" (ByVal
crColor&)

' ペン・ブラシ・フォント・ビットマップ・リージョン・パレットのいずれかの論理オブジェクトを削除し、関連付けられていた全てのシステムリソースを解放
Declare Function Api_DeleteObject& Lib "gdi32" Alias "DeleteObject" (ByVal hObject&)

' マウスのキャプチャを解放
Declare Function Api_ReleaseCapture& Lib "user32" Alias "ReleaseCapture" ()

' ウィンドウにメッセージを送信。この関数は、指定したウィンドウのウィンドウプロシージャが処理を終了するまで制御を返さない
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

#define HTCAPTION 2                         'タイトルバーをクリックしたことを示す
#define WM_NCLBUTTONDOWN &HA1               '非クライアント領域で左マウスボタンを押す

Var Shared Button1 As Object
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var lf As LOGFONT
    Var rFont As Long
    Var hDC As Long
    Var hRgn As Long
    Var txt As String
    Var Ret As Long

    txt = "札幌市白石区"
    hDC = Api_GetDC(GethWnd)                'デバイスコンテキスト取得

    lf.lfCharSet = 128                      '日本語(SHIFTJIS_CHARSET)
    lf.lfEscapement = 0                     '角度設定
```

```
    lf.lfHeight = 50                            '文字高さ設定(Point)
    lf.lfItalic = 1                             'イタリック(0/1)

    rFont = Api_CreateFontIndirect(lf)          '論理フォントの作成
    Ret = Api_SelectObject(hDC, rFont)          '指定されたデバイスコンテキストのオブジェクトを選択
    Ret = Api_BeginPath(hDC)                    'hDCで指定されたデバイスコンテキストのパスの作成
    Ret = Api_TextOut(hDC, 0, 0, txt, Len(txt))     'フォームの指定位置文字列を描画
    Ret = Api_EndPath(hDC)                      'BeginPathで開始したパスの作成を終了

    hRgn = Api_PathToRegion(hDC)                'hDCで指定したパスをリージョンに変換
    Ret = Api_SetWindowRgn(GethWnd, hRgn, True)     '指定の領域をウィンドウ領域として設定

    Ret = Api_DeleteObject(hRgn)                '論理オブジェクトの削除、システムリソースの解放
    Ret = Api_ReleaseDC(GethWnd, hDC)           'デバイスコンテキストの解放
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_MouseDown edecl (Button%, Shift%, x!,y!)
Sub MainForm_MouseDown(Button%, Shift%, x!,y!)
    Var Ret As Long

    Ret = Api_ReleaseCapture()
    Ret = Api_SendMessage(GethWnd, WM_NCLBUTTONDOWN, HTCAPTION, 0)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_DblClick edecl ()
Sub MainForm_DblClick()
    End
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

選択されているパスを線分の集合に変換(Ⅰ)

FlattenPath デバイスコンテキストに選択されているパスのすべての曲線を、一連の直線に変換
BeginPath hDCで指定されたデバイスコンテキストのパスの作成
Ellipse 楕円の描画
EndPath BeginPathで開始したパスの作成を終了
GetPath 指定されたデバイスコンテキスト内で選択されたパス内の直線の端点および曲線の制御点を定義する座標を取得
GetDC デバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放

領域を指定して楕円を描画しています。その楕円のパスを線分群に変換し、更に線上を基点とした乱数を発生させて点を描画しています。



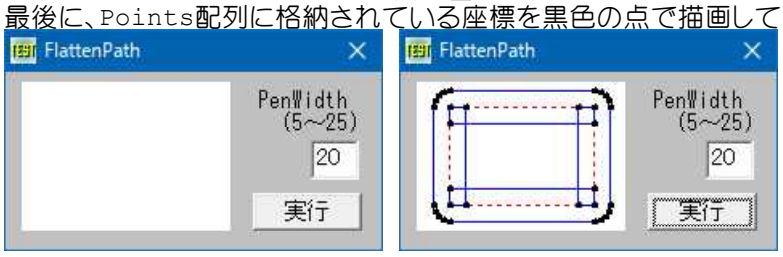
```
'=====================================
'= 選択されているパスを線分の集合に変換(Ⅰ)
'=    (FlattenPath.bas)
'=====================================
#include "Windows.bi"

Type POINTAPI
    x   As Long
    y   As Long
End Type

' 現在のデバイスコンテキストに選択されているパスのすべての曲線を、一連の直線に変換
Declare Function Api_FlattenPath& Lib "gdi32" Alias "FlattenPath" (ByVal hDC&)

' hDCで指定されたデバイスコンテキストのパスの作成
Declare Function Api_BeginPath& Lib "gdi32" Alias "BeginPath" (ByVal hDC&)

' 楕円の描画
Declare Function Api_Ellipse& Lib "gdi32" Alias "Ellipse" (ByVal hDC&, ByVal X1&, ByVal
Y1&, ByVal X2&, ByVal Y2&)

' BeginPathで開始したパスの作成を終了
Declare Function Api_EndPath& Lib "gdi32" Alias "EndPath" (ByVal hDC&)

' 指定されたデバイスコンテキスト内で選択されたパス内の直線の端点および曲線の制御点を定義する座標を取得
Declare Function Api_GetPath& Lib "gdi32" Alias "GetPath" (ByVal hDC&, ByRef Points As
POINTAPI, ByRef Types As Byte, ByVal PointNum&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

Var Shared Button1 As Object
Var Shared Picture1 As Object

Picture1.Attach GetDlgItem("Picture1")
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Static PI As Single
    Var hDC As Long
    Var PointNum As Long
    Var pa(10000) As POINTAPI
    Var Types(10000) As Byte
    Var i As Long
    Var j As Long
    Var x As Long
    Var y As Long
    Var a As Single
    Var b As Single
    Var r As Single
    Var Ret As Long

    PI = 3.14159
    hDC = Api_GetDC(Picture1.GethWnd)

    'パス作成
    Ret = Api_BeginPath(hDC)
    Ret = Api_Ellipse(hDC, 10, 10, 150, 90)
    Ret = Api_EndPath(hDC)

    '元のEllipseを確認
    Ret = Api_Ellipse(hDC, 10, 10, 150, 90)
```

```
    '線分の集合に変換
    Ret = Api_FlattenPath(hDC)

    '線分の座標を取得
    PointNum = Api_GetPath(hDC, pa(0), Types(0), 10000)

    If PointNum = 0 Then
        Z% = MessageBox("", "座標を取得できません！", 0, 2)
        Exit Sub
    End If

    '点描
    For i = 0 To PointNum - 2
        For j = 1 To 20
            a = Rnd
            b = Rnd * PI * 2
            r = Rnd * 10
            x = pa(i).x * a + pa(i + 1).x * (1 - a) + r * Cos(b)
            y = pa(i).y * a + pa(i + 1).y * (1 - a) + r * Sin(b)
            Picture1.PSet(x, y), 5
        Next
    Next
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

選択されているパスを線分の集合に変換(II)

GetPath 選択されたパス内の直線の端点および曲線の制御点を定義する座標を取得
BeginPath hDCで指定されたデバイスコンテキストのパスの作成
EndPath BeginPathで開始したパスの作成を終了
PolyDraw 直線やベジエ曲線群の描画
ExtCreatePen 指定されたスタイル、幅、ブラシ属性を持つペンを作成
SetGraphicsMode 指定されたデバイスコンテキストのグラフィックスモードを設定
GetDC デバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放

PenWidthを設定し「実行」をクリックすると、矩形領域パスを作成し、パス内直線の端点座標を取得、Points配列に格納します。
矩形領域を赤色のドットで描画し、間を置いて指定されたペン幅で青色のベジェ曲線を描画します。
最後に、Points配列に格納されている座標を黒色の点で描画しています。




```
'=======================================
'= 選択されているパスを線分の集合に変換(II)
'=    (FlattenPath2.bas)
'=======================================
#include "Windows.bi"

Type POINTAPI
    X   As Long
    Y   As Long
End Type
```

```
Type LOGBRUSH
    lbStyle    As Long
    lbColor    As Long
    lbHatch    As Long
End Type

' 長方形の描画
Declare Function Api_Rectangle& Lib "gdi32" Alias "Rectangle" (ByVal hDC&, ByVal X1&,
ByVal Y1&, ByVal X2&, ByVal Y2&)

' 論理ペンを作成
Declare Function Api_CreatePen& Lib "gdi32" Alias "CreatePen" (ByVal nPenStyle&, ByVal
nWidth&, ByVal crColor&)

' 指定されたデバイスコンテキストのオブジェクトを選択
Declare Function Api_SelectObject& Lib "gdi32" Alias "SelectObject" (ByVal hDC&, ByVal
hObject&)

' 論理オブジェクトを削除し、そのオブジェクトに関連付けられていたすべてのシステムリソースを解放
Declare Function Api_DeleteObject& Lib "gdi32" Alias "DeleteObject" (ByVal hObject&)

' 指定されたデバイスコンテキストの現在ペンを使ってパスが描かれている場合、そのパスを塗りつぶしの対象領域
として再定義
Declare Function Api_WidenPath& Lib "gdi32" Alias "WidenPath" (ByVal hDC&)

' 現在のデバイスコンテキストに選択されているパスのすべての曲線を、一連の直線に変換
Declare Function Api_FlattenPath& Lib "gdi32" Alias "FlattenPath" (ByVal hDC&)

' 指定されたデバイスコンテキスト内で選択されたパス内の直線の端点および曲線の制御点を定義する座標を取得
Declare Function Api_GetPath& Lib "gdi32" Alias "GetPath" (ByVal hDC&, ByRef Points As
POINTAPI, ByRef Types As Byte, ByVal PointNum&)

' hDCで指定されたデバイスコンテキストのパスの作成
Declare Function Api_BeginPath& Lib "gdi32" Alias "BeginPath" (ByVal hDC&)

' BeginPathで開始したパスの作成を終了
Declare Function Api_EndPath& Lib "gdi32" Alias "EndPath" (ByVal hDC&)

' 直線やベジエ曲線群の描画
Declare Function Api_PolyDraw& Lib "gdi32" Alias "PolyDraw" (ByVal hDC&, lppt As
POINTAPI, lpbTypes As byte, ByVal cCount&)

' 指定されたスタイル、幅、ブラシ属性を持つペンを作成
Declare Function Api_ExtCreatePen& Lib "gdi32" Alias "ExtCreatePen" (ByVal dwPenStyle&,
ByVal dwWidth&, ByRef lplb As LOGBRUSH, ByVal dwStyleCount&, ByRef lpStyle&)

' 指定されたデバイスコンテキストのグラフィックスモードを設定
Declare Function Api_SetGraphicsMode& Lib "gdi32" Alias "SetGraphicsMode" (ByVal hDC&,
ByVal iMode&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

#define PS_COSMETIC 0                    'コスメティックペン(幅は常に1で純色)を作成
#define PS_DASH 1                        '破線のペンを作成(--------)
#define PS_DASHDOT 3                     '一点鎖線のペンを作成(-·-·-·-)
#define PS_DASHDOTDOT 4                  '二点鎖線のペンを作成(-··-··-)
#define PS_DOT 2                         '点線のペンを作成(·····)
#define PS_ENDCAP_FLAT &H200             '端点キャップを平らにする
#define PS_ENDCAP_ROUND &H0              '端点キャップを丸くする
#define PS_ENDCAP_SQUARE &H100           '端点キャップを四角にする
#define PS_GEOMETRIC &H10000             'ジオメトリックペン(幅は任意でﾊﾟﾀｰﾝが使用できる)を作成
#define PS_INSIDEFRAME 6                 '塗りつぶし
#define PS_JOIN_BEVEL &H1000             '結合部分が平ら(ベベル接合)
#define PS_JOIN_MITER &H2000             '接合がSetMiterLimit関数で設定した範囲内にあるとき、マイター接合(結合部分が尖る)
#define PS_JOIN_ROUND &H0                '結合部分が丸くなる(ラウンド結合)
```

```
#define PS_NULL 5                               '空のペンを作成。描画は行われない
#define PS_SOLID 0                              '実線のペンを作成
#define PS_USERSTYLE 7                          'ユーザ定義鎖線(WindowsNTのみ)
#define GM_ADVANCED &H2                         'WinNT・2K・XPグラフィックモード
#define GM_COMPATIBLE &H1                       '16ビットのWindowsと互換性があるグラフィックモード

#define vbBlue &HFF0000                         '青のカラーコード
#define vbGreen &H00FF00                        '緑のカラーコード
#define vbRed &H0000FF                          '赤のカラーコード

Var Shared Text1 As Object
Var Shared Edit1 As Object
Var Shared Picture1 As Object
Var Shared Button1 As Object

Picture1.Attach GetDlgItem("Picture1")
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Text1.SetWindowText "PenWidth" & Chr$(13, 10) & "  (5～25)"
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hDC As Long
    Var hPen As Long
    Var hOldPen As Long
    Var PointNum As Long
    Var lb As LOGBRUSH
    Var OldGM As Long
    Var i As Integer
    Var Ret As Long

    Static rcX1 As Long : rcX1 = 20
    Static rcY1 As Long : rcY1 = 15
    Static rcX2 As Long : rcX2 = 110
    Static rcY2 As Long : rcY2 = 75
    Static PenWidth As Long

    Picture1.Cls

    'ペンの幅を設定
    PenWidth = Val(Edit1.GetWindowtext)

    'ピクチャボックスのDC取得
    hDC = Api_GetDC(Picture1.GethWnd)

    'グラフィックモードを設定
    OldGM = Api_SetGraphicsMode(hDC, GM_ADVANCED)

    'ペン幅を設定
    hPen = Api_ExtCreatePen(PS_GEOMETRIC Or PS_SOLID Or PS_ENDCAP_SQUARE Or
PS_JOIN_ROUND, PenWidth, lb, 0, ByVal 0)
    hOldPen = Api_SelectObject(hDC, hPen)

    '矩形パスを設定
    Ret = Api_BeginPath(hDC)
    Ret = Api_Rectangle(hDC, rcX1, rcY1, rcX2, rcY2)
    Ret = Api_EndPath(hDC)
```

```
    '現在のペンを使ってパスを拡張
    Ret = Api_WidenPath(hDC)
    Ret = Api_FlattenPath(hDC)

    'ペンを選択
    Ret = Api_SelectObject(hDC, hOldPen)
    Ret = Api_DeleteObject(hPen)

    'DCからデータパスを取得
    PointNum = Api_GetPath(hDC, ByVal 0, ByVal 0, 0)

    If (PointNum) Then
        Var Points(PointNum - 1) As POINTAPI
        Var Types(PointNum - 1) As Byte

        Ret = Api_GetPath(hDC, Points(0), Types(0), PointNum)
    End If

    'ペンの色を赤に設定
    hPen = Api_CreatePen(PS_DOT, 0, vbRed)
    hOldPen = Api_SelectObject(hDC, hPen)

    '矩形領域を赤で描画
    Ret = Api_Rectangle(hDC, rcX1, rcY1, rcX2, rcY2)
    Ret = Api_SelectObject(hDC, hOldPen)
    Ret = Api_DeleteObject(hPen)

    Wait 300

    'ペンの色を青に設定
    hPen = Api_CreatePen(PS_SOLID, 1, vbBlue)
    hOldPen = Api_SelectObject(hDC, hPen)

    'ベジエ曲線群の描画
    Ret = Api_PolyDraw(hDC, Points(0), Types(0), PointNum)
    Ret = Api_SelectObject(hDC, hOldPen)
    Ret = Api_DeleteObject(hPen)

    Wait 300

    'FlattenPathで取得した線分群の座標点を描画
    Picture1.SetDrawWidth 4
    For i = 0 To PointNum - 1
        Picture1.Pset(Points(i).X, Points(i).Y)
    Next

    'グラフィックモードを元に戻す
    Ret = Api_SetGraphicsMode(hDC, OldGM)

    'デバイスコンテキストの解放
    Ret = Api_ReleaseDC(Picture1.GethWnd, hDC)
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

前方部分検索･全体一致検索

前方部分検索･全体一致検索･指定文字列検索
SendMessage 指定のウィンドウにメッセージを送る
LB_FINDSTRING(&H18F) 前方部分検索

LB_FINDSTRINGEXACT(&H1A2) 全体一致検索

エディットボックスに入力した文字列からリストボックス内の文字列を検索し項目を選択します。
前方部分検索 LB_FINDSTRING　　全体一致検索 LB_FINDSTRINGEXACT


```
'========================================
'= 前方部分検索･全体一致検索
'=    (SendMessage.bas)
'========================================
#include "Windows.bi"

' ウィンドウにメッセージを送信。この関数は、指定したウィンドウのウィンドウプロシージャが処理を終了するまで制御を返さない
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal wMsg&, ByVal wParam&, lParam As Any)

#define LB_FINDSTRING &H18F                 '前方部分検索
#define LB_FINDSTRINGEXACT &H1A2            '全体一致検索

Var Shared Edit1 As Object
Var Shared List1 As Object
Var Shared Radio(1) As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
For i = 0 To 1
    Radio(i).Attach GetDlgItem("Radio" & Trim$(Str$(i + 1)))
    Radio(i).SetFontSize 14
Next i

'========================================
'=
'========================================
Declare Function Index bdecl () As Integer
Function Index()
    Index = Val(Mid$(GetDlgRadioSelect("Radio1"), 6)) - 1
End Function

'========================================
'=
'========================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    List1.AddString "Computer"
    List1.AddString "Screen"
    List1.AddString "Modem"
    List1.AddString "Printer"
    List1.AddString "Scanner"
    List1.AddString "Sound Blaster"
    List1.AddString "Keyboard"
    List1.AddString "CD-ROM"
    List1.AddString "Mouse"

    Edit1.SetFocus
End Sub
```

```
'=====================================
'=
'=====================================
Declare Sub Edit1_Change edecl ()
Sub Edit1_Change()
    Var Msg As Long
    Var Ret As Integer

    If Index = 0 Then
        Msg = LB_FINDSTRING
    Else
        Msg = LB_FINDSTRINGEXACT
    End If

    Ret = Api_SendMessage(List1.GethWnd, Msg, -1, Edit1.GetWindowText)
    List1.SetCursel Ret
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

ソートテスト

例では、データ数を100000以内で用意し、乱数を発生させ ソートしています。


```
'=====================================
'= ソートテスト
'=    (Sort2.bas)
'=====================================
#include "Windows.bi"

Var Shared Edit1 As Object
Var Shared Text1 As Object
Var Shared List(1) As Object
Var Shared Button1 As Object

For i = 0 To 1
    List(i).Attach GetDlgItem("List" & Trim$(Str$(i + 1))) : List(i).SetFontSize 14
Next
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
```

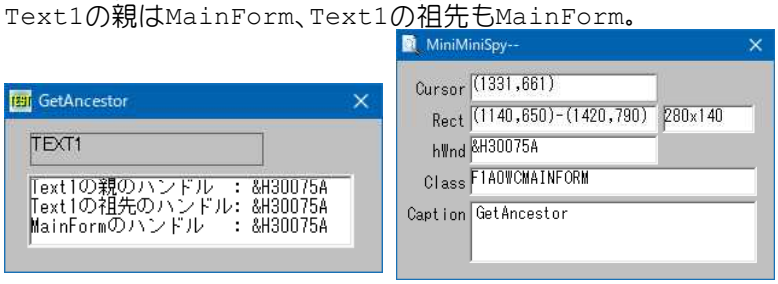
```
Var Shared dMax As Long
Var Shared d(100000) As Long

'=====================================
'= Sort部
'=====================================
Declare Sub Sort ()
Sub Sort()
    If dMax Mod 2 = 0 Then SF = dMax Else SF = dMax + 1
    Dim Ls(SF/2), Rs(SF/2)

    k = 0 : l = 1 : r = dMax
    *Jp0
    i = l : j = r : t = d((l + r) / 2)
    *Jp1
    If d(i) < t Then i = i + 1 : Goto *Jp1
    *Jp2
    If t < d(j) Then j = j - 1 : Goto *Jp2
    If i < j Then Swap d(i), d(j) : i = i + 1 : j = j - 1 : Goto *Jp1
    If i = j Then i = i + 1 : j = j - 1
    If l >= j Then *Jp3
    If i < r Then Ls(k) = l : Rs(k) = j : k = k + 1 : l = i : Goto *Jp0
     r = j : Goto *Jp0
    *Jp3
    If i < r Then l = i : Goto *Jp0
    k = k - 1
    If k >= 0 Then l = Ls(k) : r = Rs(k) : Goto *Jp0

    Erase LS, RS
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    dMax = Val(GetDlgItemText("Edit1"))
    If dMax < 2 Or dMax > 100000 Then
        Edit1.SetWindowText ""
        Edit1.SetFocus
        Exit Sub
    End If

    SetMousePointer 2
    List(0).Resetcontent
    List(1).Resetcontent

    For CT = 1 To dMax
        d(CT) = Rnd(1) * dMax
        List(0).AddString Format$(d(CT)," ###,###")
    Next

    Sort

    For CT = 1 To dMax
        List(1).AddString Format$(d(CT)," ###,###")
    Next

    SetMousePointer 0
End Sub

'=====================================
'=
'=====================================
Declare Sub Edit1_Change edecl ()
Sub Edit1_Change()
    List(0).Resetcontent
    List(1).Resetcontent
End Sub
```

```
'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

祖先のハンドルを取得

指定したウィンドウの祖先のハンドルを取得します。
GetAncestor 指定したウィンドウの祖先のハンドルを取得
GetParent 指定された子ウィンドウの親ウィンドウまたはオーナーウィンドウのハンドルを返す

Text1の祖先のハンドル、Text1の親のハンドル、メインフォームのハンドルを取得しています。
Text1の親はMainForm、Text1の祖先もMainForm。



MiniMiniSpy--でMainFormのハンドルを取得(&H5C035E)

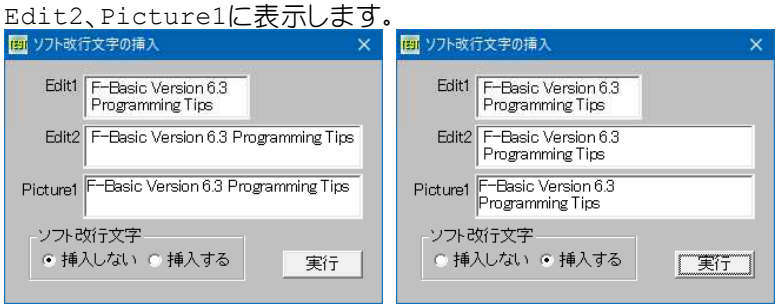
```
'=====================================
'= 祖先のハンドルを取得
'=    (GetAncestor.bas)
'=====================================
#include "Windows.bi"

' 指定したウィンドウの祖先のハンドルを取得
Declare Function Api_GetAncestor& Lib "user32" Alias "GetAncestor" (ByVal hWnd&, ByVal
gaFlags&)

' 指定された子ウィンドウの親ウィンドウまたはオーナーウィンドウのハンドルを返す
Declare Function Api_GetParent& Lib "user32" Alias "GetParent" (ByVal hWnd&)

#define GA_PARENT 1                         '親ウィンドウを返す
#define GA_ROOT 2                           '親子関係を遡って、直近上位のトップレベルウィンドウを返
                                            す
#define GA_ROOTOWNER 3                      '親子関係と所有関係を遡って、所有されていないトップレベ
                                            ルウィンドウを返す
#define vbCrLf (Chr$(13) & Chr$(10))        'キャリッジリターンとラインフィード(¥r¥n)

Var Shared Text1 As Object
Var Shared Text2 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Text2.Attach GetDlgItem("Text2") : Text2.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var hWnd1 As Long
    Var hWnd2 As Long
    Var txt As String

    hWnd1 = Api_GetAncestor(Text1.GethWnd, GA_ROOT)
    hWnd2 = Api_GetParent(Text1.GethWnd)

    txt = txt & "Text1の親のハンドル　: &&H" & Hex$(hWnd2) & vbCrLf
```

```
    txt = txt & "Text1の祖先のハンドル: &&H" & Hex$(hWnd1) & vbCrLf
    txt = txt & "MainFormのハンドル    : &&H" & Hex$(GethWnd)
    Text2.SetWindowText txt
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

ソフト改行文字の挿入

SendMessage ウィンドウにメッセージを送信
EM_FMTLINES(&HC8) フト改行文字の設定をオンまたはオフ

#define EM_FMTLINES &HC8 は複数行入力ありに設定したエディットボックスにおいて、ソフト改行文字を設定または削除します。
wParam = 1 ソフト改行文字を挿入/wParam = 0 ソフト改行文字を挿入しない。lParam は常に0
例では、F-Basic Version 6.3 Programming Tips の文字列が入れきらない幅のEdit1、および全表示可能な幅のEdit2、Picture1を用意します。
EditBoxは複数行入力あり、垂直スクロールありに設定しています。(Picture1はリソース操作で3D表示にしています)
起動時(通常)Edit1には文字列が入りきらないので2行表示されます。
ソフト改行文字を挿入するをチェックし「実行」をクリックするとEdit1の改行位置を取得しソフト改行文字を挿入してEdit2、Picture1に表示します。

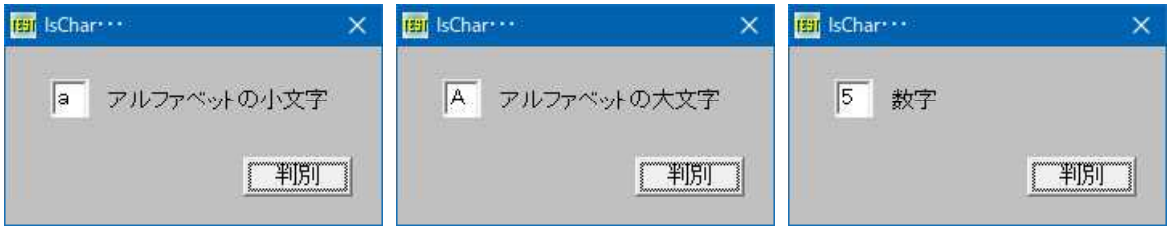



```
'=====================================
'= ソフト改行文字の挿入
'=    (EM_FMTLINES.bas)
'=====================================
#include "Windows.bi"

' ウィンドウにメッセージを送信。この関数は、指定したウィンドウのウィンドウプロシージャが処理を終了するまで制御を返さない
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal wMsg&, ByVal wParam&, lParam As Any)

#define EM_FMTLINES &HC8                    'ソフト改行文字の設定をオンまたはオフにする

Var Shared Edit(1) As Object
Var Shared Text(2) As Object
Var Shared Radio(1) As Object
Var Shared Picture1 As Object
Var Shared Button1 As Object
Var Shared Group1 As Object

For i = 0 To 2
    If i < 2 Then
        Edit(i).Attach GetDlgItem("Edit" & Trim$(Str$(i + 1))) : Edit(i).SetFontSize 14
        Radio(i).Attach GetDlgItem("Radio" & Trim$(Str$(i + 1))) : Radio(i).SetFontSize 14
    End If
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1))) : Text(i).SetFontSize 14
```

```
Next
Picture1.Attach GetDlgItem("Picture1")
Button1.Attach GetDlgItem("button1") : Button1.SetFontSize 14
Group1.Attach GetDlgItem("Group1") : Group1.SetFontSize 14

Var Shared txt As String

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    ShowWindow -1

    txt = "F-Basic Version 6.3 Programming Tips"
    Edit(0).SetWindowText txt
    Edit(1).SetWindowText txt
    Picture1.Cls
    Picture1.Print txt
End Sub

'=====================================
'=
'=====================================
Declare Function Index bdecl () As Integer
Function Index()
    Index = Val(Mid$(GetDlgRadioSelect("Radio1"), 6)) - 1
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    Ret = Api_SendMessage(Edit(0).GethWnd, EM_FMTLINES, CLng(Index), ByVal CLng(0))

    txt = Edit(0).GetWindowText

    Picture1.Cls
    Picture1.Print txt

    Edit(1).SetWindowText txt
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

対象一文字が数字・アルファベットの大文字・小文字

例ではエディットボックスに一文字入力し、数字・アルファベットの大文字・アルファベットの小文字の判別をしています。
IsCharAlpha 指定された文字が、アルファベット文字かを判別
IsCharAlphaNumeric 指定された文字が、数字かどうかを判別
IsCharLower 指定された文字が、小文字かどうかを判別
IsCharUpper 指定された文字が、大文字かどうかを判別

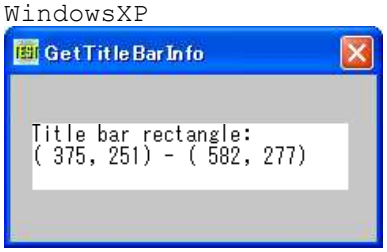



```
'=====================================
'=  対象一文字が数字･アルファベットの大文字･小文字
'=    (IsChar.bas)
'=====================================
#include "Windows.bi"

' 指定された文字が、アルファベット文字かを判断
Declare Function Api_IsCharAlpha& Lib "user32" Alias "IsCharAlphaA" (ByVal cChar As byte)

' 指定された文字が、数字かどうかを判断
Declare Function Api_IsCharAlphaNumeric& Lib "user32" Alias "IsCharAlphaNumericA" (ByVal
cChar As byte)

' 指定された文字が、小文字かどうかを判断
Declare Function Api_IsCharLower& Lib "user32" Alias "IsCharLowerA" (ByVal cChar As byte)

' 指定された文字が、大文字かどうかを判断
Declare Function Api_IsCharUpper& Lib "user32" Alias "IsCharUpperA" (ByVal cChar As byte)

Var Shared Edit1 As Object
Var Shared Text1 As Object
Var Shared Button1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var sChr As String

    If Edit1.GetWindowText = "" Then Exit Sub
    If Api_IsCharAlphaNumeric(Asc(Left$(Edit1.GetWindowText, 1))) Then sChr = "数字"
    If Api_IsCharAlpha(Asc(Left$(Edit1.GetWindowText, 1))) Then sChr = "アルファベット"
    If Api_IsCharLower(Asc(Left$(Edit1.GetWindowText, 1))) Then sChr = sChr & "の小文字"
    If Api_IsCharUpper(Asc(Left$(Edit1.GetWindowText, 1))) Then sChr = sChr & "の大文字"

    Text1.SetWindowText sChr
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

タイトルバー(長方形)の情報取得

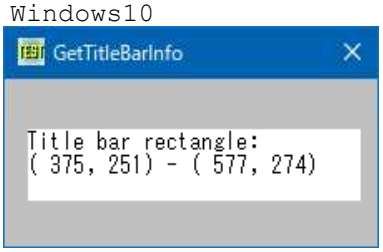
タイトルバーの表示部分(長方形)を取得します。
GetTitleBarInfo タイトルバーの情報を取得

WindowsXP



クラシックスタイル

Windows10



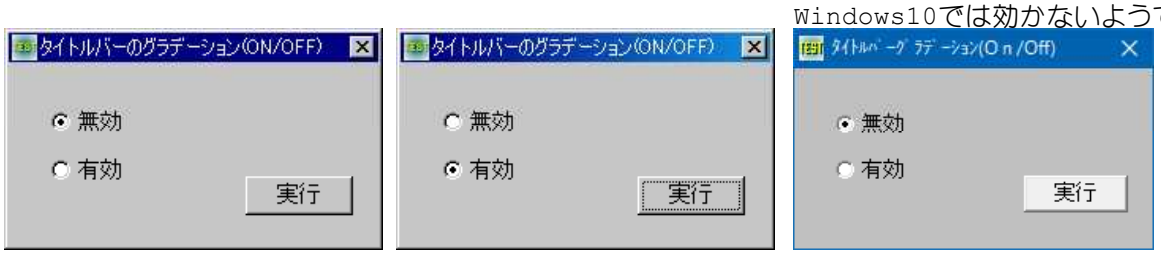

```
'=====================================
'= タイトルバー(長方形)情報取得
'=    (GetTitleBarInfo.bas)
'=====================================
#include "Windows.bi"

#define STATE_SYSTEM_FOCUSABLE &H100000   'フォーカス可能
#define STATE_SYSTEM_INVISIBLE &H8000     '非表示
#define STATE_SYSTEM_OFFSCREEN &H10000    '非可視状態
#define STATE_SYSTEM_PRESSED &H8          '押下状態
#define STATE_SYSTEM_UNAVAILABLE &H1      '無効
#define CCHILDREN_TITLEBAR 5              '

Type RECT
    Left       As Long
    Top        As Long
    Right      As Long
    Bottom     As Long
End Type

Type TITLEBARINFO
    cbSize                          As Long
    rcTitleBar                      As RECT
    rgstate(CCHILDREN_TITLEBAR)     As Long
End Type

' タイトルバーの表示部分(長方形)を取得
Declare Function Api_GetTitleBarInfo& Lib "user32" Alias "GetTitleBarInfo" (ByVal hWnd&,
ByRef pti As TITLEBARINFO)

Var Shared Text1 As Object
Var Shared Timer1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Timer1.Attach getDlgItem("Timer1")

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var ti As TITLEBARINFO
    Var Ret As Long

    '構造体初期化
    ti.cbSize = Len(ti)

    'このウィンドウのタイトルバー情報を取得
    Ret = Api_GetTitleBarInfo(GethWnd, ti)

    '情報を表示
    Text1.SetWindowText "Title bar rectangle:" & Chr$(13) & "(" &
Str$(ti.rcTitleBar.Left) & "," & Str$(ti.rcTitleBar.Top) & ") - (" &
Str$(ti.rcTitleBar.Right) & "," & Str$(ti.rcTitleBar.Bottom) & ")"
End Sub

'=====================================
'=
'=====================================
While 1
```

```
    WaitEvent
Wend
Stop
End
```

タイトルバーのグラデーション効果

タイトルバーのグラデーション効果をON/OFFします。
SystemParametersInfo システム全体に関するパラメータを取得・設定
SPI_SETGRADIENTCAPTIONS(&H1009) システムに関するパラメータを示す
SPIF_UPDATEINIFILE(&H1) ユーザープロファイルの更新を指定
SPIF_SENDWININICHANGE(&H2) すべてのトップレベルウィンドウに変更を通知

例では、WindowsXPクラシックでテストしています。Windows2000・Windows9xも同様です。
Windows10では効かないようです。




```
'=====================================
'= タイトルバーのグラデーション効果
'=    (SetGradientCaptions.bas)
'=====================================
#include "Windows.bi"

#define SPI_SETGRADIENTCAPTIONS &H1009     'システムに関するパラメータを示す定数の宣言
#define SPIF_UPDATEINIFILE &H1             'ユーザープロファイルの更新を指定する定数の宣言
#define SPIF_SEndWININICHANGE &H2          'すべてのトップレベルウィンドウに変更を通知する定数宣言
#define SPIF_SEndCHANGE SPIF_SEndWININICHANGE

' システム全体に関するパラメータを取得・設定
Declare Function Api_SystemParametersInfo& Lib "user32" Alias "SystemParametersInfoA"
(ByVal uiAction&, ByVal uiParam&, pvParam As Any, ByVal fWinIni&)

'=====================================
'=
'=====================================
Declare Function Index bdecl () As Long
Function Index()
    Index = Val(Mid$(GetDlgRadioSelect("Radio1"), 6)) - 1
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    Ret = Api_SystemParametersInfo(SPI_SETGRADIENTCAPTIONS, 0, ByVal Index,
SPIF_UPDATEINIFILE Or SPIF_SEndCHANGE)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

タイトルバーのないフォームの移動

フォームの移動はタイトルバーをドラッグしますが、ない場合のフォームを移動させる場合便利です。
ReleaseCapture マウスのキャプチャを解放する
SendMessage ウィンドウにメッセージを送る

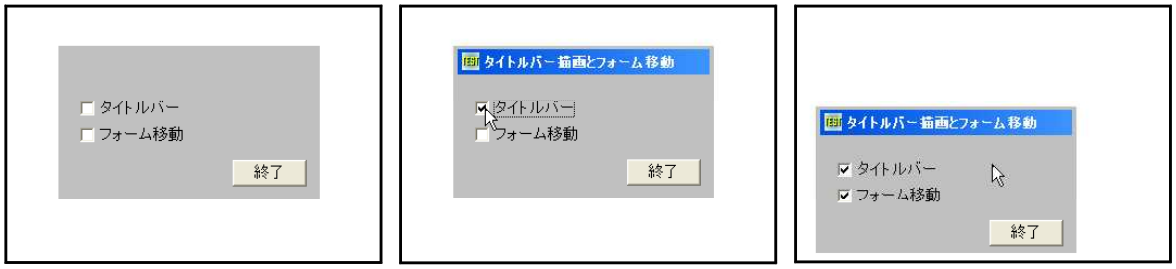
タイトルバーの有無にかかわらず機能しますが、例ではなしに設定しています。
フォーム内をドラッグして移動させます。コントロール上はドラッグできません。終了はフォームを ダブルクリックしてください。

```
'========================================
'= タイトルバーのないフォームの移動
'=    (ReleaseCapture2.bas)
'========================================
#include "Windows.bi"

' ウィンドウにメッセージを送信
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

' マウスのキャプチャを解放
Declare Function Api_ReleaseCapture& Lib "user32" Alias "ReleaseCapture" ()

#define WM_NCLBUTTONDOWN &HA1                 '非クライアント領域で左マウスボタンを押す
#define HTCAPTION 2                           'タイトルバーをクリックしたことを示す
#define vbLeftButton 1                        '左ボタンクリック

'========================================
'=
'========================================
Declare Sub MainForm_MouseMove edecl (ByVal Button As Integer, ByVal Shift As Integer,
ByVal SX As Single, ByVal SY As Single)
Sub MainForm_MouseMove(ByVal Button As Integer, ByVal Shift As Integer, ByVal SX As
Single, ByVal SY As Single)
    Var Ret As Long

    If Button = 1 Then
        Ret = Api_ReleaseCapture
        Ret = Api_SendMessage(GethWnd, WM_NCLBUTTONDOWN, HTCAPTION, 0)
    End If
End Sub

'========================================
'=
'========================================
Declare Sub MainForm_DblClick edecl (ByVal Button As Integer, ByVal Shift As Integer,
ByVal SX As Single, ByVal SY As Single)
Sub MainForm_DblClick(ByVal Button As Integer, ByVal Shift As Integer, ByVal SX As Single,
ByVal SY As Single)
    End
End Sub

'========================================
'=
'========================================
While 1
    WaitEvent
Wend
Stop
End
```

タイトルバーの描画とフォーム移動

DrawCaption 指定のウィンドウのキャプションを指定のデバイスコンテキストに描画
DrawFrameControl 指定されたタイプとスタイルを備える、ボタンやスクロールバーなどのフレームコントロールを描画
SetRect RECT構造体の値を設定
SendMessage ウィンドウにメッセージを送信
ReleaseCapture マウスのキャプチャを解放
GetDC 指定されたウィンドウのデバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放

通常のプロパティ設定では、アイコンを表示できないためリソースを下記のとおり書き換えています。
CREATE3, XWT_FORM, 0x80080115L, 0x0L
(カットアンドトライで目的を達成できただけであり、自己責任で…)



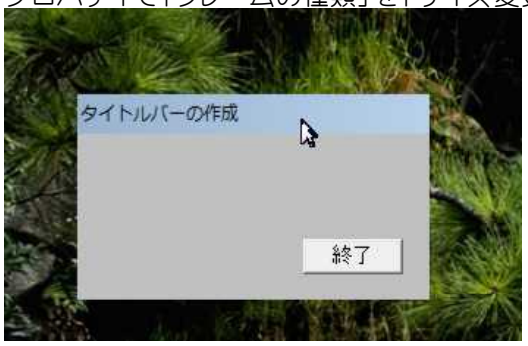

```
'=====================================
'= タイトルバーの描画とフォーム移動
'=   (DrawFrameControl6.bas)
'=====================================
#include "Windows.bi"

Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

' 指定のウィンドウのキャプションを指定のデバイスコンテキストに描画
Declare Function Api_DrawCaption& Lib "user32" Alias "DrawCaption" (ByVal hWnd&, ByVal
hDC&, pcRect As RECT, ByVal un&)

' 指定されたタイプとスタイルを備える、ボタンやスクロールバーなどのフレームコントロールを描画
Declare Function Api_DrawFrameControl& Lib "user32" Alias "DrawFrameControl" (ByVal
hDC&, lpRect As RECT, ByVal un1&, ByVal un2&)

' RECT構造体の値を設定
Declare Function Api_SetRect& Lib "user32" Alias "SetRect" (lpRect As RECT, ByVal X1&,
ByVal Y1&, ByVal X2&, ByVal Y2&)

' ウィンドウにメッセージを送信
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

' マウスのキャプチャを解放
Declare Function Api_ReleaseCapture& Lib "user32" Alias "ReleaseCapture" ()

' 指定されたウィンドウのデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

#define DC_ACTIVE 1                     'アクティブなときの色
#define DC_ICON 4                       'アイコン付き
#define DC_TEXT 8                       'タイトルテキスト付き
#define DC_GRADIENT &H20                'グラデーション
#define WM_NCLBUTTONDOWN &HA1           '非クライアント領域で左マウスボタンを押す
#define HTCAPTION 2                     'タイトルバーをクリックしたことを示す
```

```
Var Shared Check1 As Object
Var Shared Check2 As Object
Var Shared Button1 As Object

Check1.Attach GetDlgItem("Check1") : Check1.SetFontSize 14
Check2.Attach GetDlgItem("Check2") : Check2.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Form_Paint ()
Sub Form_Paint()
    Var rc As RECT
    Var hDC As Long
    Var Ret As Long

    hDC = Api_GetDC(GethWnd)

    Cls

    If Check1.GetCheck = 1 Then
        Ret = Api_SetRect(rc, 4, 3, GetWidth - 4, 26)
        Ret = Api_DrawCaption(GethWnd, hDC, rc, DC_ACTIVE Or DC_ICON Or DC_TEXT Or
DC_GRADIENT)
    End If

    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_MouseMove edecl (ByVal Button As Integer, ByVal Shift As Integer,
ByVal X As Single, ByVal Y As Single)
Sub MainForm_MouseMove(ByVal Button As Integer, ByVal Shift As Integer, ByVal X As Single,
ByVal Y As Single)
    Var Ret As Long

    If Button = 1 And Check2.GetCheck = 1 Then
        Ret = Api_ReleaseCapture
        Ret = Api_SendMessage(GethWnd, WM_NCLBUTTONDOWN, HTCAPTION, 0)
    End If
End Sub

'=====================================
'=
'=====================================
Declare Sub Check1_on edecl ()
Sub Check1_on()
    Form_Paint
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    End
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

タイトルバーを作成する

タイトルバーを作成してみます。
DrawCaption 指定のウィンドウのキャプションを指定のデバイスコンテキストに描画
SetRect RECT構造体の値を設定
SendMessage ウィンドウにメッセージを送信
ReleaseCapture マウスのキャプチャを解放

フォームのコントロールを「なし」に設定。作成したタイトルバーをドラッグして移動させます。(コントロールを「あり」とするだけでいいのでは？と、突っ込まないで…)
プロパティで「フレームの種類」を「サイズ変更(太枠)」にすると右図のようになります。チョット変ですね。




```
'========================================
'= タイトルバーを作成する
'=    (MakeTitleBar.bas)
'========================================
#include "Windows.bi"

Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

' 指定のウィンドウのキャプションを指定のデバイスコンテキストに描画
Declare Function Api_DrawCaption& Lib "user32" Alias "DrawCaption" (ByVal hWnd&, ByVal
hDC&, pcRect As RECT, ByVal un&)

' RECT構造体の値を設定
Declare Function Api_SetRect& Lib "user32" Alias "SetRect" (lpRect As RECT, ByVal X1&,
ByVal Y1&, ByVal X2&, ByVal Y2&)

' ウィンドウにメッセージを送信。この関数は、指定したウィンドウのウィンドウプロシージャが処理を終了するまで制御を返さない
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

' マウスのキャプチャを解放
Declare Function Api_ReleaseCapture& Lib "user32" Alias "ReleaseCapture" ()

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

#define WS_BORDER &H800000                  'フォームの枠線がある
#define WM_NCLBUTTONDOWN &HA1               '非クライアント領域で左マウスボタンを押す
#define HTCAPTION 2                         'タイトル バーをクリックしたことを示す
#define DC_ACTIVE 1                         'アクティブなときの色
#define DC_BINS 6                           'プリンタで使用できる給紙方法を取得
#define DC_COPIES 18                        '最大部数を取得
#define DC_GRADIENT &H20                    'グラデーション
#define DC_ICON 4                           'アイコン付き
#define DC_INBUTTON &H10                    'ボタンとして描画
#define DC_NOTACTIVE &H2                    '
```

```
#define DC_SMALLCAP 2                           '幅の狭いタイプ
#define DC_TEXT 8                               'タイトルテキスト付き
#define TitleHeight 26                          'タイトルバーの高さ

Var Shared Button1 As Object

Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var Shared rct As RECT
Var Shared hDC As Long

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var Ret As Long

    hDC = Api_GetDC(GethWnd)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_MouseDown edecl (ByVal Button As Integer, ByVal Shift As Integer,
ByVal x As single, ByVal y As single)
Sub MainForm_MouseDown(ByVal Button As Integer, ByVal Shift As Integer, ByVal x As single,
ByVal y As single)
    If y > TitleHeight Then Exit Sub
    Var Ret As Long

    Ret = Api_ReleaseCapture()
    Ret = Api_SendMessage(GethWnd, WM_NCLBUTTONDOWN, HTCAPTION, 0)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Resize edecl ()
Sub MainForm_Resize()
    Var Ret As Long

    Cls
    Ret = Api_SetRect(rct, 0, 0, GetWidth, TitleHeight)
    Ret = Api_DrawCaption(GethWnd, hDC, rct, DC_ACTIVE Or DC_TEXT Or DC_GRADIENT)

    If GetWidth < 240 Or GetHeight < 140 Then SetWindowSize 240, 140
    Button1.MoveWindow GetWidth - 86, GetHeight - 42
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    Ret = Api_ReleaseDC(GethWnd, hDC)
    End
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

タイムゾーンの取得と変更

Windows9x系と、WindowsNT系ではレジストリの位置が異なるため、バージョンにより切り替えています。
GetTimeZoneInformation タイムゾーン情報を取得
SetTimeZoneInformation タイムゾーン情報を設定
RegOpenKeyEx レジストリのキーのハンドルを確保
RegQueryValueEx (RegQueryValueExString)レジストリの値を取得
RegEnumKey 指定された開いているレジストリーキーのサブキーを列挙
RegCloseKey レジストリのハンドルを解放
MultiByteToWideChar ANSI文字列をUnicode文字列に変換
CopyMemory ある位置から別の位置にメモリブロックを移動

例では、起動時に世界のタイムゾーンがListBoxに表示され、ダブルクリックした項目のタイムゾーンに切り替わる状態を表しています。
「日付と時刻のプロパティ」は、表示された状態で切り替わるものではありません。


```
'=======================================
'= タイムゾーンの取得と変更
'=(SetTimeZoneInformation.bas)
'=======================================
#include "Windows.bi"

#define VER_PLATFORM_WIN32_NT 2             'WINDOWSNT、2000、XP
#define VER_PLATFORM_WIN32_WINDOWS 1        'WINDOWS9x

Type OSVERSIONINFO
    dwOSVersionInfoSize     As Long
    dwMajorVersion          As Long
    dwMinorVersion          As Long
    dwBuildNumber           As Long
    dwPlatformId            As Long
    szCSDVersion            As String * 128
End Type

' オペレーティングシステムの種類やバージョンに関する情報を取得
Declare Function Api_GetVersionEx& Lib "Kernel32" Alias "GetVersionExA"
(lpVersionInformation As OSVERSIONINFO)

Type SYSTEMTIME
    wYear               As Integer
    wMonth              As Integer
    wDayOfWeek          As Integer
    wDay                As Integer
    wHour               As Integer
    wMinute             As Integer
    wSecond             As Integer
    wMilliseconds       As Integer
End Type

Type REGTIMEZONEINFORMATION
    Bias                As Long
    StandardBias        As Long
    DaylightBias        As Long
    StandardDate        As SYSTEMTIME
    DaylightDate        As SYSTEMTIME
End Type

Type TIME_ZONE_INFORMATION
    Bias As Long
    StandardName(63)  As Byte               'ユニコード文字列取得のため使用
    StandardDate      As SYSTEMTIME
```

```
    StandardBias      As Long
    DaylightName(63)  As Byte                'ユニコード文字列取得のため使用
    DaylightDate      As SYSTEMTIME
    DaylightBias      As Long
End Type

#define TIME_ZONE_ID_INVALID &HFFFFFFFF
#define TIME_ZONE_ID_UNKNOWN 0
#define TIME_ZONE_ID_STANDARD 1
#define TIME_ZONE_ID_DAYLIGHT 2

' タイムゾーン情報を取得
Declare Function Api_GetTimeZoneInformation& Lib "kernel32" Alias
"GetTimeZoneInformation" (lpTimeZoneInformation As TIME_ZONE_INFORMATION)

' タイムゾーン情報を設定
Declare Function Api_SetTimeZoneInformation& Lib "kernel32" Alias
"SetTimeZoneInformation" (lpTimeZoneInformation As TIME_ZONE_INFORMATION)

#define REG_SZ 1                              'ヌル終端文字列
#define REG_BINARY 3                          'バイナリデータ
#define REG_DWORD 4                           '32ビットの数値
#define HKEY_CLASSES_ROOT -2147483648         '拡張子に関する情報や、それらとアプリケーションとの関連
                                              づけに関する情報
#define HKEY_CURRENT_USER -2147483647         '現在Windowsにログインしているユーザーの情報
#define HKEY_LOCAL_MACHINE -2147483646        'PCを利用するユーザーに共通の設定情報
#define HKEY_USERS -2147483645                'Windowsを利用するユーザー個別の情報

#define ERROR_SUCCESS 0                       '正常終了の戻り値を示す
#define ERROR_BADDB 1                         '
#define ERROR_BADKEY 2                        '
#define ERROR_CANTOPEN 3                      '
#define ERROR_CANTREAD 4                      '
#define ERROR_CANTWRITE 5                     '
#define ERROR_OUTOFMEMORY 6                   '
#define ERROR_ARENA_TRASHED 7                 '
#define ERROR_ACCESS_DENIED 8                 '
#define ERROR_INVALID_PARAMETERS 87           '
#define ERROR_NO_MORE_ITEMS 259               '
#define KEY_ALL_ACCESS &H3F                   'レジストリに対するすべての権限を許可
#define REG_OPTION_NON_VOLATILE 0             '

' レジストリのキーのハンドルを確保
Declare Function Api_RegOpenKeyEx& Lib "advapi32" Alias "RegOpenKeyExA" (ByVal hKey&,
ByVal lpSubKey$, ByVal ulOptions&, ByVal samDesired&, phkResult&)

' レジストリの値を取得
Declare Function Api_RegQueryValueEx& Lib "advapi32" Alias "RegQueryValueExA" (ByVal
hKey&, ByVal lpszValueName$, ByVal lpdwReserved&, lpdwType&, lpData As Any, lpcbData&)

' レジストリの値を取得
Declare Function Api_RegQueryValueExString& Lib "advapi32" Alias "RegQueryValueExA"
(ByVal hKey&, ByVal lpValueName$, ByVal lpReserved&, lpType&, ByVal lpData$, lpcbData&)

' 指定された開いているレジストリーキーのサブキーを列挙
Declare Function Api_RegEnumKey& Lib "advapi32" Alias "RegEnumKeyA" (ByVal hKey&, ByVal
dwIndex&, ByVal lpName$, ByVal cbName&)

' レジストリのハンドルを解放
Declare Function Api_RegCloseKey& Lib "advapi32" Alias "RegCloseKey" (ByVal hKey&)

#define SKEY_NT "SOFTWARE¥Microsoft¥Windows NT¥CurrentVersion¥Time Zones"
#define SKEY_9X "SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥Time Zones"

' ANSI文字列をUnicode文字列に変換
Declare Function Api_MultiByteToWideChar& Lib "kernel32" Alias "MultiByteToWideChar"
(ByVal CodePage&, ByVal dwFlags&, lpMultiByteStr As Any, ByVal cchMultiByte&,
lpWideCharStr As Any, ByVal cchWideChar&)

#define CP_ACP 0
```

```
#define MB_PRECOMPOSED &H1

' ある位置から別の位置にメモリブロックを移動
Declare Sub CopyMemory Lib "Kernel32" Alias "RtlMoveMemory" (Destination As Any, Source
As Any, ByVal Length&)

Var Shared List1 As Object

List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
List1.SetWindowSize 210, 90

Var Shared SubKey As String

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var CurIdx As Long
    Var DataLen As Long
    Var ValueLen As Long
    Var hKey As Long
    Var strvalue As String
    Var osV As OSVERSIONINFO
    Var Ret As Long

    osV.dwOSVersionInfoSize = Len(osV)
    Ret = Api_GetVersionEx(osV)
    If osV.dwPlatformId = VER_PLATFORM_WIN32_NT Then
        SubKey = SKEY_NT
    Else
        SubKey = SKEY_9X
    End If

    Ret = Api_RegOpenKeyEx(HKEY_LOCAL_MACHINE, SubKey, 0, KEY_ALL_ACCESS, hKey)

    If Ret = ERROR_SUCCESS Then
        CurIdx = 0
        DataLen = 32
        ValueLen = 32
        Do
            strvalue = String$(ValueLen, 0)
            Ret = Api_RegEnumKey(hKey, CurIdx, strvalue, DataLen)

            If Ret = ERROR_SUCCESS Then
                List1.AddString Left$(strvalue, ValueLen)
            End If
            CurIdx = CurIdx + 1
        Loop While Ret = ERROR_SUCCESS
        Ret = Api_RegCloseKey(hKey)
    Else
        List1.AddString "Could not open registry key"
    End If
End Sub

'=====================================
'=
'=====================================
Declare Sub List1_DblClick edecl ()
Sub List1_DblClick()
    Var TZ As TIME_ZONE_INFORMATION
    Var oldTZ As TIME_ZONE_INFORMATION
    Var rTZI As REGTIMEZONEINFORMATION
    Var bytDLTName(32) As Byte
    Var bytSTDName(32) As Byte
    Var cbStr As Long
    Var dwType As Long
    Var hKey As Long
    Var lngData As Long
    Var Ret As Long
```

```
    Ret = Api_RegOpenKeyEx(HKEY_LOCAL_MACHINE, SubKey & "¥" &
List1.GetText(List1.GetCursel), 0, KEY_ALL_ACCESS, hKey)

    If Ret = ERROR_SUCCESS Then
        Ret = Api_RegQueryValueEx(hKey, "TZI", 0&, ByVal 0&, rTZI, Len(rTZI))
        If Ret = ERROR_SUCCESS Then
            TZ.Bias = rTZI.Bias
            TZ.StandardBias = rTZI.StandardBias
            TZ.DaylightBias = rTZI.DaylightBias
            TZ.StandardDate = rTZI.StandardDate
            TZ.DaylightDate = rTZI.DaylightDate
            cbStr = 32
            dwType = REG_SZ

            Ret = Api_RegQueryValueEx(hKey, "Std", 0, dwType, bytSTDName(0), cbStr)

            If Ret = ERROR_SUCCESS Then
                Ret = Api_MultiByteToWideChar(CP_ACP, MB_PRECOMPOSED, bytSTDName(0),
cbStr, TZ.StandardName(0), 32)
            Else
                Ret = Api_RegCloseKey(hKey)
                Exit Sub
            End If

            cbStr = 32
            dwType = REG_SZ

            Ret = Api_RegQueryValueEx(hKey, "Dlt", 0, dwType, bytDLTName(0), cbStr)

            If Ret = ERROR_SUCCESS Then
                Ret = Api_MultiByteToWideChar(CP_ACP, MB_PRECOMPOSED, bytDLTName(0),
cbStr, TZ.DaylightName(0), 32)
            Else
                Ret = Api_RegCloseKey(hKey)
                Exit Sub
            End If

            Ret = Api_GetTimeZoneInformation(oldTZ)

            If Ret = TIME_ZONE_ID_INVALID Then
                A% = MessageBox("", "Error getting original TimeZone Info", 0, 2)
                Ret = Api_RegCloseKey(hKey)
                Exit Sub
            Else
                If TZ.DaylightDate.wMonth <> 0 And TZ.DaylightBias <> 0 Then
                    Ret = Api_SetTimeZoneInformation(TZ)
                Else
                    CopyMemory TZ.DaylightName(0), TZ.StandardName(0), 64
                    TZ.DaylightBias = 0
                    Ret = Api_SetTimeZoneInformation(TZ)
                End If
                A% = MessageBox("", "タイムゾーンを変更しました！" & Chr$(13, 10) &  "「OK」クリック
で元に戻ります。", 0, 2)
                Ret = Api_SetTimeZoneInformation(oldTZ)
            End If
        End If

        Ret = Api_RegCloseKey(hKey)
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

ダイヤラを呼び出す

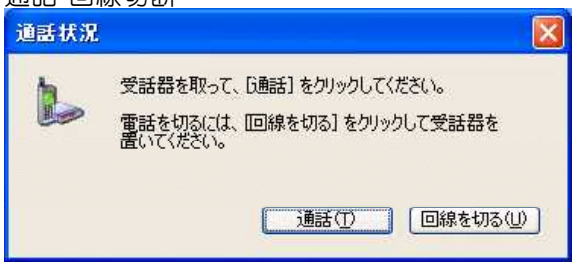
Windows附属のダイヤラを利用します。
tapiRequestMakeCall TAPI32(ダイヤラ)を呼び出す

電話番号を入力し「ダイヤラ」をクリックします。

WindowsXPでのダイヤラ接続画面

Windows2000でのダイヤラ接続画面

通話・回線切断

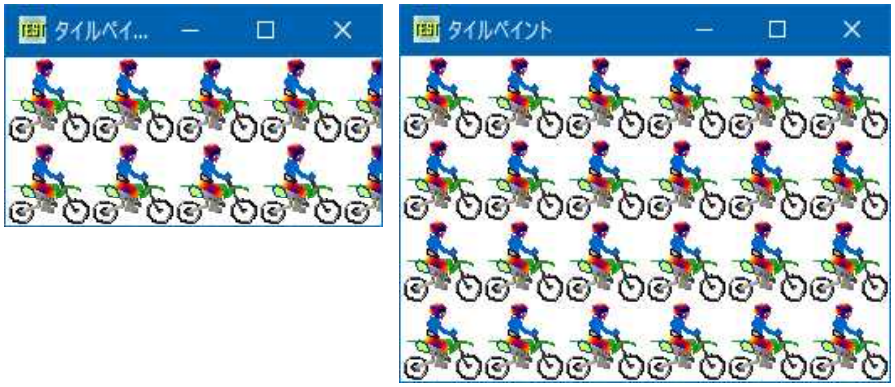



```
'=======================================
'= テレホンダイヤラー
'=    (TelDialer.bas)
'=======================================
#include "Windows.bi"

' TAPI32(ダイヤラー)を呼び出す
Declare Function Api_tapiRequestMakeCall& Lib "TAPI32" Alias "tapiRequestMakeCall"
(ByVal DestAddress$, ByVal AppName$, ByVal CalledParty$, ByVal Comment$)

#define TAPIERR_NOREQUESTRECIPIENT -2
#define TAPIERR_REQUESTQUEUEFULL -3
#define TAPIERR_INVALDESTADDRESS -4

Var Shared Edit1 As Object
Var Shared Button1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=======================================
'=
'=======================================
Declare Sub DialNumber(Number As String, sName As String)
Sub DialNumber(Number As String, sName As String)
    Var Buff As String
    Var Ret As Long

    sName = "TOKO"
    Ret = Api_tapiRequestMakeCall(Trim$(Number), Chr$(0), Trim$(sName), "")

    If Ret <> 0 Then
        Buff = "ダイヤルエラー№ : "

        select case Ret
```

```
            case TAPIERR_NOREQUESTRECIPIENT
                Buff = Buff & "Windowsのダイヤラーが起動できません！"

            case TAPIERR_REQUESTQUEUEFULL
                Buff = Buff & "ダイヤル要求が満杯です！"

            case TAPIERR_INVALDESTADDRESS
                Buff = Buff & "指定の電話番号はありません！"

            case Else
                 Buff = Buff & "未知のエラーです！"

        End select

        A% = MessageBox("", Buff, 0, 2)
    End If
End Sub

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Number As String

    Number = Edit1.GetWindowText()

    DialNumber Number, "Test"
End Sub

'=======================================
'=
'=======================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
    End
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

タイルペイント(Ⅰ)

BitBlt コピー元からコピー先のデバイスコンテキストへ、指定された長方形内の各ピクセルの色データをコピー
GetDC デバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放



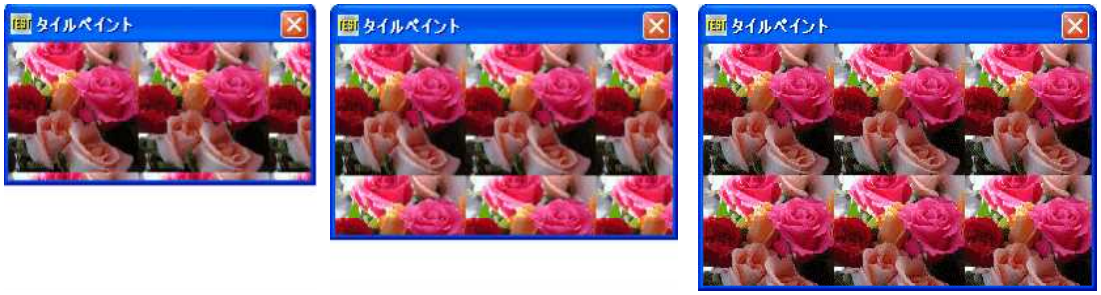
フォームをリサイズ

```
'=====================================
'= タイルペイント
'=    (TilePaint.bas)
'=====================================
#include "Windows.bi"

' ビットブロック転送を行う。コピー元からコピー先のデバイスコンテキストへ、指定された長方形内の各ピクセルの色データをコピー
Declare Function Api_BitBlt& Lib "gdi32" Alias "BitBlt" (ByVal hDestDC&, ByVal X&, ByVal Y&, ByVal nWidth&, ByVal nHeight&, ByVal hSrcDC&, ByVal xSrc&, ByVal ySrc&, ByVal dwRop&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

#define SRCCOPY &HCC0020                    'そのまま転送

Var Shared Bitmap As Object
Var Shared Picture1 As Object

Picture1.Attach GetDlgItem("Picture1")
BitmapObject Bitmap

Var Shared hDC As Long
Var Shared hDC2 As Long

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Bitmap.LoadFile "bike1.Bmp"
    Picture1.StretchBitmap Bitmap, 0, 0, Picture1.GetWidth, Picture1.GetHeight
    Bitmap.DeleteObject

    hDC = Api_GetDC(GethWnd)
    hDC2 = Api_GetDC(Picture1.GethWnd)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Resize edecl ()
Sub MainForm_Resize()
    Var y As Long
    Var x As Long
    Var Ret As Long

    Cls

    For y = 0 To GetHeight Step Picture1.GetHeight
        For x = 0 To GetWidth Step Picture1.GetWidth
            Ret = Api_BitBlt(hDC, x, y, Picture1.GetWidth, Picture1.GetHeight, hDC2, 0, 0, SRCCOPY)
```

```
        Next
    Next
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
    Ret = Api_ReleaseDC(GethWnd, hDC)
    Ret = Api_ReleaseDC(Picture1.GethWnd, hDC2)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

タイルペイント(II)

BitBlt コピー元からコピー先のデバイスコンテキストへ、指定された長方形内の各ピクセルの色データをコピー
LoadImage 画像ファイルの読み込み
CreateCompatibleDC デバイスコンテキストを作成
DeleteDC 指定されたデバイスコンテキストを削除
SelectObject 指定されたデバイスコンテキストのオブジェクトを選択
DeleteObject 論理オブジェクトを削除し、そのオブジェクトに関連付けられていたすべてのシステムリソース解放
GetObject オブジェクト取得
GetDC デバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放



フォームをリサイズ

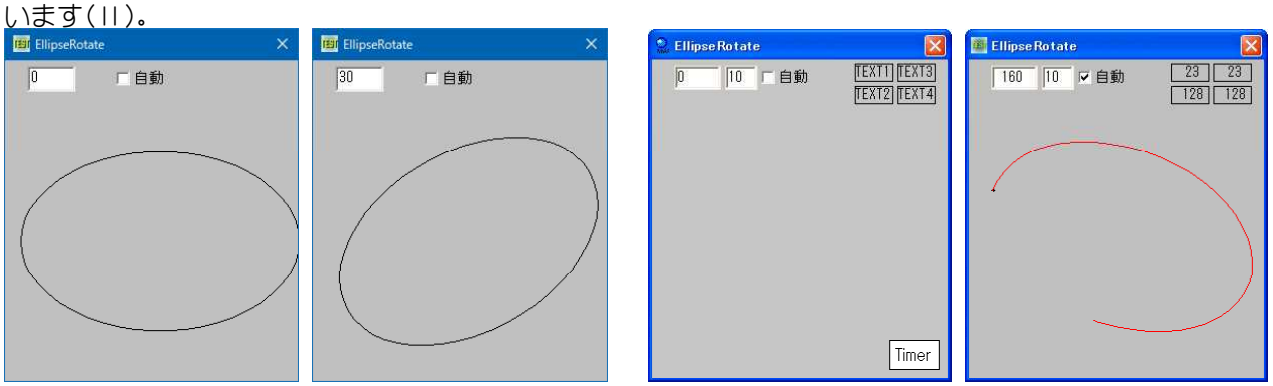
```
'=====================================
'= タイルペイント(II)
'=    (TilePaint2.bas)
'=====================================
#include "Windows.bi"

Type BITMAP
    bmType          As Long
    bmWidth         As Long
    bmHeight        As Long
    bmWidthBytes    As Long
    bmPlanes        As Integer
    bmBitsPixel     As Integer
    bmBits          As Long
End Type

#define IMAGE_BITMAP 0                  'ビットマップ
#define LR_LOADFROMFILE &H10            '外部ファイルからロードする
#define SRCCOPY &HCC0020                'そのまま転送

' 画像ファイルの読み込み
Declare Function Api_LoadImage& Lib "user32" Alias "LoadImageA" (ByVal hInst&, ByVal
```

```
lpszName$, ByVal uType&, ByVal cxDesired&, ByVal cyDesired&, ByVal fuLoad&)

' ビットブロック転送を行う。コピー元からコピー先のデバイスコンテキストへ、指定された長方形内の各ピクセルの色データをコピー
Declare Function Api_BitBlt& Lib "gdi32" Alias "BitBlt" (ByVal hDestDC&, ByVal X&, ByVal
Y&, ByVal nWidth&, ByVal nHeight&, ByVal hSrcDC&, ByVal xSrc&, ByVal ySrc&, ByVal dwRop&)

' 指定されたデバイスコンテキストに関連するデバイスと互換性のあるメモリデバイスコンテキストを作成
Declare Function Api_CreateCompatibleDC& Lib "gdi32" Alias "CreateCompatibleDC" (ByVal
hDC&)

' 指定されたデバイスコンテキストを削除
Declare Function Api_DeleteDC& Lib "gdi32" Alias "DeleteDC" (ByVal hDC&)

' 指定されたデバイスコンテキストのオブジェクトを選択
Declare Function Api_SelectObject& Lib "gdi32" Alias "SelectObject" (ByVal hDC&, ByVal
hObject&)

' 論理オブジェクトを削除し、そのオブジェクトに関連付けられていたすべてのシステムリソースを解放
Declare Function Api_DeleteObject& Lib "gdi32" Alias "DeleteObject" (ByVal hObject&)

' オブジェクト取得
Declare Function Api_GetObject& Lib "gdi32" Alias "GetObjectA" (ByVal hObject&, ByVal
nCount&, lpObject As Any)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

'=====================================
'=
'=====================================
Declare Sub MainForm_Resize edecl ()
Sub MainForm_Resize()
    Var hDC As Long                         'MainFormのデバイスコンテキスト
    Var Success As Long                     'API戻り値
    Var bmp As BITMAP                       'BITMAP構造体
    Var srcDC As Long                       'ソースのデバイスコンテキスト
    Var srcBmp As Long                      'ソースビットマップ
    Var hSrcBmp As Long                     'ソースビットマップのハンドル
    Var y As Long
    Var x As Long
    Var Ret As Long

    hDC = Api_GetDC(GethWnd)

    srcBmp = Api_LoadImage(0, "flower.bmp", IMAGEBITMAP, 0, 0, LR_LOADFROMFILE)
    Ret = Api_GetObject(srcBmp, Len(bmp), bmp)
    srcDC = Api_CreateCompatibleDC(hDC)
    hSrcBmp = Api_SelectObject(srcDC, srcBmp)

    For y = 0 To GetHeight Step bmp.bmHeight
        For x = 0 To GetWidth Step bmp.bmWidth
            Ret = Api_BitBlt(hDC, x, y, bmp.bmWidth, bmp.bmHeight, srcDC, 0, 0, &HCC0020)
        Next x
    Next y

    Ret = Api_DeleteObject(hSrcBmp)
    Ret = Api_DeleteDC(srcDC)
    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
```

```
Stop
End
```

楕円形の回転(Ⅰ)(Ⅱ)

楕円形を描画し回転させてみます。
EditBoxに直接角度を入力するとその角度で楕円を描画します。自動にチェックを入れると0～350°まで自動的に回転させています。
フォームをリサイズすると楕円の大きさも変わります。
左:角度の間隔を設定して自動的に回転させています(Ⅰ)。右図:「自動」のチェックを外して指定の角度で回転させています(Ⅱ)。



描画スピードを調整し状況を確認しています。リストは楕円の回転(Ⅱ)

```
'=====================================
'= 楕円形の回転(Ⅰ)
'=   (EllipseRotate.bas)
'=====================================
#include "Windows.bi"

Var Shared Edit1 As Object
Var Shared Edit2 As Object
Var Shared Check1 As Object
Var Shared Timer1 As Object

Edit1.Attach GetDlgItem("EDit1") : Edit1.SetFontSize 14
Edit2.Attach GetDlgItem("EDit2") : Edit2.SetFontSize 14
Check1.Attach GetDlgItem("Check1") : Check1.SetFontSize 14
Timer1.Attach GetDlgItem("Timer1")

Var Shared mCx As Single
Var Shared mCy As Single
Var Shared mWidth As Single
Var Shared mHeight As Single
Var Shared eStep As Integer

#define PI 3.14159265

'=====================================
'=
'=====================================
Declare Sub DrawEllipse(cx As Single, cy As Single, wid As Single, hgt As Single, Angle As
Single)
Sub DrawEllipse(cx As Single, cy As Single, wid As Single, hgt As Single, Angle As Single)
    Var Sin_Angle As Single
    Var Cos_Angle As Single
    Var theta As Single
    Var dtheta As Single
    Var X As Single
    Var Y As Single
    Var RX As Single
    Var RY As Single

    Cls
```

```
    Angle = Angle * PI / 180
    Sin_Angle = Sin(Angle)
    Cos_Angle = Cos(Angle)

    theta = 0
    dtheta = 2 * PI / 50

    '最初の位置
    X = wid * Cos(theta)
    Y = hgt * Sin(theta)

    RX = cx + X * Cos_Angle + Y * Sin_Angle
    RY = cy - X * Sin_Angle + Y * Cos_Angle
    Pset(RX, RY)

    Do While theta < 2 * PI
        theta = theta + dtheta
        X = wid * Cos(theta)
        Y = hgt * Sin(theta)

        RX = cx + X * Cos_Angle + Y * Sin_Angle
        RY = cy - X * Sin_Angle + Y * Cos_Angle

        Line - (RX, RY)
    Loop
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Timer1.SetInterval 5
    Timer1.Enable 0
    Edit2.ShowWindow 0
End Sub

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var Angle As Single

    On Error Goto *Er_Trap

    Angle = Val(Edit1.GetWindowText)
    eStep = Val(Edit2.GetWindowText)
    Angle = Angle + eStep
    If Angle > 350 Then Angle = 0
    Edit1.SetWindowText Str$(Angle)
    Exit Sub

*Er_Trap
    Resume Next
End Sub

'=====================================
'=
'=====================================
Declare Sub Edit1_Change edecl ()
Sub Edit1_Change()
    Var Angle As Single

    On Error Goto *Er_Trap
    Angle = Val(Edit1.GetWindowText)
    On Error Goto 0

    DrawEllipse mCx, mCy, mWidth, mHeight, Angle
    Exit Sub
```

```
*Er_Trap
    Resume Next
End Sub

'=====================================
'=
'=====================================
Declare Sub Edit1_SetFocus edecl ()
Sub Edit1_SetFocus()
    Timer1.Enable 0
    Edit2.ShowWindow 0
    Check1.SetCheck 0
End Sub

'=====================================
'=
'=====================================
Declare Sub Check1_on edecl ()
Sub Check1_on()
    If CHeck1.GetCheck = 0 Then
        Timer1.Enable 0
        Edit2.ShowWindow 0
        eStep = 0
    Else
        Timer1.Enable -1
        Edit2.ShowWindow -1
    End If
    SetFocus
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_ReSize edecl ()
Sub MainForm_ReSize()
    mCx = GetWidth / 2
    mCy = GetHeight / 2

    If GetWidth > GetHeight Then
        mWidth = GetHeight * 0.45
        mHeight = GetWidth * 0.25
    Else
        mWidth = GetWidth * 0.45
        mHeight = GetHeight * 0.25
    End If

    Edit1_Change
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

```
'=====================================
'= 楕円形の回転(II)
'=    (EllipseRotate2.bas)
'=====================================
#include "Windows.bi"

Var Shared Text1 As Object
Var Shared Text2 As Object
```

```
Var Shared Text3 As Object
Var Shared Text4 As Object
Var Shared Edit1 As Object
Var Shared Edit2 As Object
Var Shared Check1 As Object
Var Shared Timer1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Text2.Attach GetDlgItem("Text2") : Text2.SetFontSize 14
Text3.Attach GetDlgItem("Text3") : Text3.SetFontSize 14
Text4.Attach GetDlgItem("Text4") : Text4.SetFontSize 14
Edit1.Attach GetDlgItem("EDit1") : Edit1.SetFontSize 14
Edit2.Attach GetDlgItem("EDit2") : Edit2.SetFontSize 14
Check1.Attach GetDlgItem("Check1") : Check1.SetFontSize 14
Timer1.Attach GetDlgItem("Timer1")

Var Shared mCx As Single
Var Shared mCy As Single
Var Shared mWidth As Single
Var Shared mHeight As Single
Var Shared eStep As Integer

#define PI 3.14159265

'=====================================
'= cx:楕円中心(x) cy:楕円中心(y) wid:フォーム幅 hgt:フォーム高さ angle:角度
'=====================================
Declare Sub DrawEllipse(cx As Single, cy As Single, wid As Single, hgt As Single, angle As
Single)
Sub DrawEllipse(cx As Single, cy As Single, wid As Single, hgt As Single, angle As Single)
    Var sin_angle As Single                 'サイン
    Var cos_angle As Single                 'コサイン
    Var theta As Single                     'θ
    Var dtheta As Single                    'θ
    Var X As Single
    Var Y As Single
    Var RX As Single                        '楕円描画の点(X)
    Var RY As Single                        '楕円描画の点(Y)

    Cls

    angle = angle * PI / 180
    sin_angle = Sin(angle)
    cos_angle = Cos(angle)

    theta = 0
    dtheta = 2 * PI / 50                    '楕円を描くライン数(50の線で描画)

    '最初の位置
    X = wid * Cos(theta)
    Y = hgt * Sin(theta)

    RX = cx + X * cos_angle + Y * sin_angle
    RY = cy - X * sin_angle + Y * cos_angle

    SetDrawWidth 3                          '描画する太さ(3)
    Pset(RX, RY), 0                         '楕円描画の始まり(黒の点)
    SetDrawWidth 0                          '描画する太さ(通常に戻す)
    Text1.SetWindowText Str$(Int(RX))       '点のX位置
    Text2.SetWindowText Str$(Int(RY))       '点のY位置


    Do While theta < 2 * PI
        theta = theta + dtheta
        X = wid * Cos(theta)
        Y = hgt * Sin(theta)

        RX = cx + X * cos_angle + Y * sin_angle
        RY = cy - X * sin_angle + Y * cos_angle
```

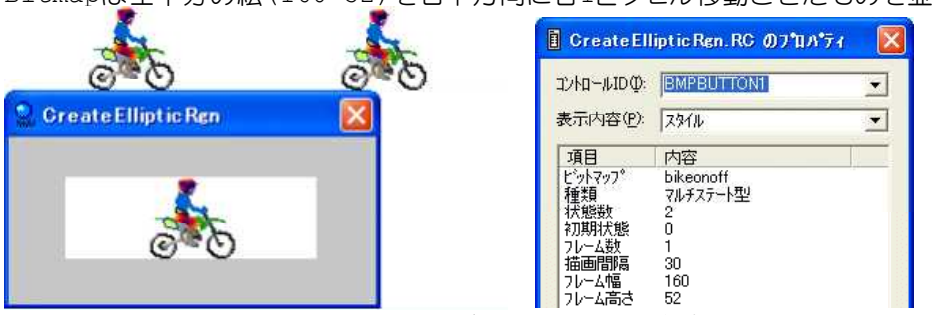
```
        Line - (RX, RY), , 5
        Text3.SetWindowText Str$(Int(RX))
        Text4.SetWindowText Str$(Int(RY))
        Wait 5
    Loop
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Timer1.SetInterVal 10
    Timer1.Enable 0
    Edit2.ShowWindow 0
End Sub

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var angle As Single

    On Error GoTo *Er_Trap

    angle = Val(Edit1.GetWindowText)        '角度
    eStep = Val(Edit2.GetWindowText)        '角度のステップ
    angle = angle + eStep
    If angle > 350 Then angle = 0           '350°まで描画 → 0°
    Edit1.SetWindowText Str$(angle)
    Exit Sub

*Er_Trap
    Resume Next
End Sub

'=====================================
'=
'=====================================
Declare Sub Edit1_Change edecl ()
Sub Edit1_Change()
    Var angle As Single

    On Error Goto *Er_Trap
    angle = Val(Edit1.GetWindowText)
    On Error GoTo 0

    DrawEllipse mCx, mCy, mWidth, mHeight, angle
    Exit Sub

*Er_Trap
    Resume Next
End Sub

'=====================================
'=
'=====================================
Declare Sub Edit1_SetFocus edecl ()
Sub Edit1_SetFocus()
    Timer1.Enable 0
    Edit2.ShowWindow 0
    Check1.SetCheck 0
End Sub

'=====================================
'=
'=====================================
Declare Sub Check1_on edecl ()
Sub Check1_on()
```

```
    If CHeck1.GetCheck = 0 Then
        Timer1.Enable 0
        Edit2.ShowWindow 0
        eStep = 0
    Else
        Timer1.Enable -1
        Edit2.ShowWindow -1
    End If
    SetFocus
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_ReSize edecl ()
Sub MainForm_ReSize()
    mCx = GetWidth / 2
    mCy = GetHeight / 2

    If GetWidth > GetHeight Then
        mWidth = GetHeight * 0.45
        mHeight = GetWidth * 0.25
    Else
        mWidth = GetWidth * 0.45
        mHeight = GetHeight * 0.25
    End If

    Edit1_Change
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

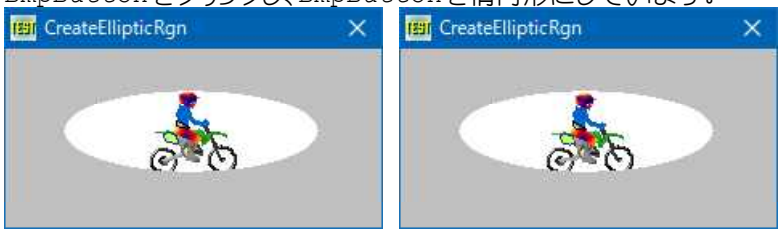
楕円形の領域を作成

楕円形の領域を作成します。
CreateEllipticRgn 円形・楕円形の領域を作成
SetWindowRgn 指定の領域をウィンドウ領域として設定

例では、BmpButtonに図のBitmap(320×52)を貼り付けています。
Bitmapは左半分の絵(160×52)を右下方向に各1ピクセル移動させたものを並べています。



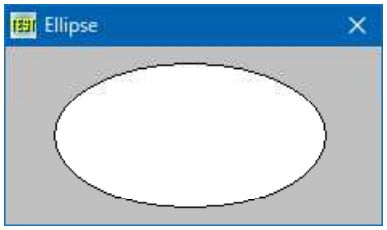
BmpButtonをクリックし、BmpButtonを楕円形にしています。



```
'=====================================
'= 楕円形の領域を作成
'=    (CreateEllipticRgn.bas)
'=====================================
#include "Windows.bi"

' 円形･楕円形の領域を作成
Declare Function Api_CreateEllipticRgn& Lib "gdi32" Alias "CreateEllipticRgn" (ByVal
X1&, ByVal Y1&, ByVal X2&, ByVal Y2&)

' 指定の領域をウィンドウ領域として設定
Declare Function Api_SetWindowRgn& Lib "user32" Alias "SetWindowRgn" (ByVal hWnd&, ByVal
hRgn&, ByVal bRedraw&)

Var Shared BmpButton1 As Object

BmpButton1.Attach GetDlgItem("BmpButton1")

'=====================================
'=
'=====================================
Declare Sub BmpButton1_on edecl ()
Sub BmpButton1_on()
    Var cw As Long
    Var ch As Long
    Var hWnd As Long
    Var hr As Long
    Var Ret As Long

    cw = BmpButton1.GetWidth
    ch = BmpButton1.GetHeight
    hWnd = BmpButton1.GethWnd
    hr = Api_CreateEllipticRgn(0, 0, cw, ch)
    Ret = Api_SetWindowRgn(hWnd, hr, -1)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

楕円形を描画

Ellipse 楕円形を描画
GetDC デバイスコンテキストハンドルを取得
ReleaseDC デバイスコンテキストを解放

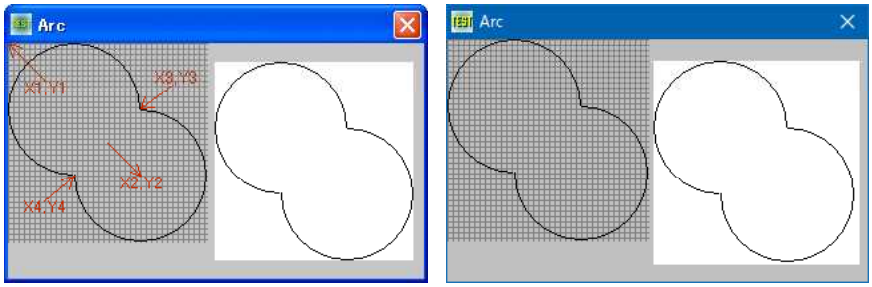



```
'=====================================
'= 楕円形を描画
'=    (Ellipse.bas)
'=====================================
#include "Windows.bi"
```

```
' 楕円の描画
Declare Function Api_Ellipse& Lib "gdi32" Alias "Ellipse" (ByVal hDC&, ByVal X1&, ByVal
Y1&, ByVal X2&, ByVal Y2&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var hDC As Long
    Var Ret As Long

    'フォームのデバイスコンテキスト取得
    hDC = Api_GetDC(GethWnd)

    '楕円の表示
    Ret = Api_Ellipse(hDC, 30, 10, 200, 100)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Qeryclose edecl ()
Sub MainForm_Qeryclose()
    Ret = ReleaseDC(hWnd, hDC)
    End
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

楕円弧を描画(Ⅰ)

フォームとPictureに楕円弧をテスト描画しています。x,yの各位置を把握するためグリッドを描いています。
Arc 楕円弧を描く




```
Declare Function Api_Arc& lib "gdi32" alias "Arc" (byval hDC&, byval X1&, byval Y1&, byval
X2&, byval Y2&, byval X3&, byval Y3&, byval X4&, byval Y4&)
```

X1 楕円に外接する長方形の水平位置
Y1 楕円に外接する長方形の垂直位置
X2 楕円に外接する長方形の水平位置
Y2 楕円に外接する長方形の垂直位置
X3 弧の始点位置の水平位置
Y3 弧の始点位置の垂直位置
X4 弧の終了位置の水平位置

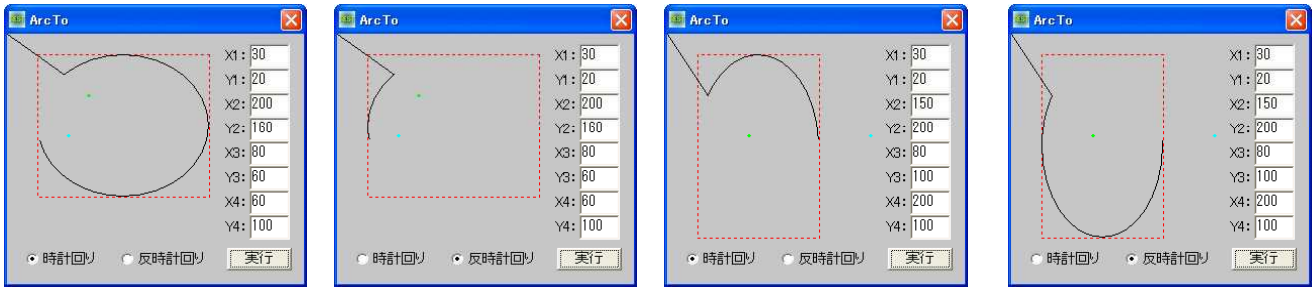
Y4 弧の終了位置の垂直位置

```
'=====================================
'= 楕円弧を描く
'=    (Arc.bas)
'=====================================
#include "Windows.bi"

' 楕円を描画
Declare Function Api_Arc& Lib "gdi32" Alias "Arc" (ByVal hDC&, ByVal X1&, ByVal Y1&, ByVal
X2&, ByVal Y2&, ByVal X3&, ByVal Y3&, ByVal X4&, ByVal Y4&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

Var Shared Picture1 As Object

Picture1.Attach GetDlgItem("Picture1")

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var hDC As Long
    Var Ret As Long

    For i = 0 To 149 Step 4
        Line(i, 0) - (i, 150), , 1
        Line(0, i) - (150, i), , 1
    Next

    'フォームに描画
    hDC = Api_GetDC(GethWnd)
    Ret = Api_Arc(hDC, 0, 0, 100, 100, 100, 50, 50, 100)
    Ret = Api_Arc(hDC, 50, 50, 150, 150, 50, 100, 100, 50)
    Ret = Api_ReleaseDC(GethWnd, hDC)

    'ピクチャボックスに描画
    hDC = Api_GetDC(Picture1.GethWnd)
    Ret = Api_Arc(hDC, 0, 0, 100, 100, 100, 50, 50, 100)
    Ret = Api_Arc(hDC, 50, 50, 150, 150, 50, 100, 100, 50)
    Ret = Api_ReleaseDC(Picture1.GethWnd, hDC)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

楕円弧の描画(II)

ArcTo 楕円形の円弧の描画

X1～X4、Y1～Y4の数値および時計回り、反時計回り描画を変えて変化を確認しています。

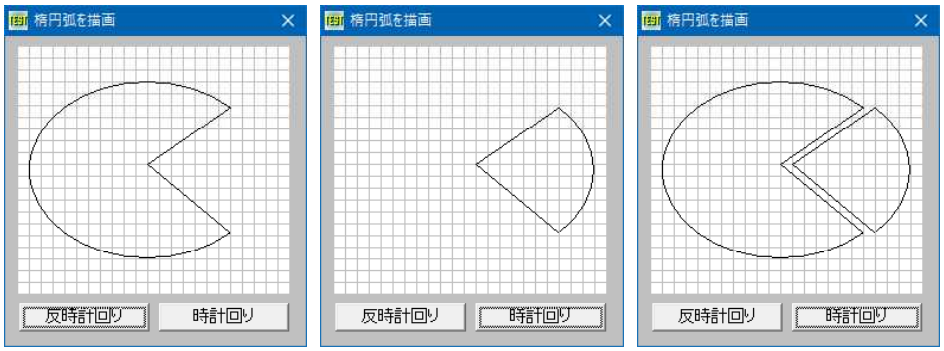



```
X1 LeftRect
Y1 TopRect
X2 RightRect
Y2 BottomRect
X3 xStartArc
Y3 yStartArc
X4 xEndArc
Y4 yEndAr

'=====================================
'= 楕円形の円弧の描画
'=    (ArcTo.bas)
'=====================================
#include "Windows.bi"

#define AD_CLOCKWISE 2                  '時計回り
#define AD_COUNTERCLOCKWISE 1           '反時計回り

' 楕円弧を描画
Declare Function Api_ArcTo& Lib "gdi32" Alias "ArcTo" (ByVal hDC&, ByVal X1&, ByVal Y1&,
ByVal X2&, ByVal Y2&, ByVal X3&, ByVal Y3&, ByVal X4&, ByVal Y4&)

' デバイスコンテキストに設定されている、円弧の現在の方向を設定
Declare Function Api_SetArcDirection& Lib "gdi32" Alias "SetArcDirection" (ByVal hDC&,
ByVal ArcDirection&)

' デバイスコンテキストに設定されている、円弧の現在の方向を取得
Declare Function Api_GetArcDirection& Lib "gdi32" Alias "GetArcDirection" (ByVal hDC&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

Var Shared x(3) As Long
Var Shared y(3) As Long

'=====================================
'=
'=====================================
Declare Function Index bdecl () As Integer
Function Index()
    Index = Val(Mid$(GetDlgRadioSelect("Radio1"), 6)) -1
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long
    Var hDC As Long

    x(0) = Val(GetDlgItemText("Edit1"))    'LeftRect
    y(0) = Val(GetDlgItemText("Edit2"))    'TopRect
    x(1) = Val(GetDlgItemText("Edit3")     'RightRect
    y(1) = Val(GetDlgItemText("Edit4"))    'BottomRect
    x(2) = Val(GetDlgItemText("Edit5"))    'xStartArc
```

```
    y(2) = Val(GetDlgItemText("Edit6"))    'yStartArc
    x(3) = Val(GetDlgItemText("Edit7"))    'xEndArc
    y(3) = Val(GetDlgItemText("Edit8"))    'yEndArc

    cls
    hDC = Api_GetDC(GethWnd)

    If Index = 0 Then
        Ret = Api_SetArcDirection(hDC, AD_CLOCKWISE)
    Else
        Ret = Api_SetArcDirection(hDC, AD_COUNTERCLOCKWISE)
    End If

    Line(x(0), y(0)) - (x(1), y(1)), , 5, b, Dot
    Ret = Api_ArcTo(hDC, x(0), y(0), x(1), y(1), x(2), y(2), x(3), y(3))
    SetDrawWidth 3
    Pset(x(2), y(2)),9
    Pset(x(3), y(3)),11
    SetDrawWidth 0
    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

楕円弧を描画（III）

グラフでよく見かけます。楕円の一部を切り取り右にずらして描画しています。
数値の位置関係を把握するため、10ドット間隔の線を引いています。
SetArcDirection デバイスコンテキストに設定されている、円弧の現在の方向を設定
ArcTo 楕円弧を描画
MoveToEx 現在位置を受け取るバッファを参照で指定
LineTo 現在の位置から終点までを直線で描画




```
'=======================================
'= 楕円弧を描画（III）
'=    (SetArcDirection.bas)
'=======================================
#include "Windows.bi"

' デバイスコンテキストに設定されている、円弧の現在の方向を設定
Declare Function Api_SetArcDirection& Lib "gdi32" Alias "SetArcDirection" (ByVal hDC&,
ByVal ArcDirection&)

' 楕円弧を描画
Declare Function Api_ArcTo& Lib "gdi32" Alias "ArcTo" (ByVal hDC&, ByVal X1&, ByVal Y1&,
ByVal X2&, ByVal Y2&, ByVal X3&, ByVal Y3&, ByVal X4&, ByVal Y4&)
```

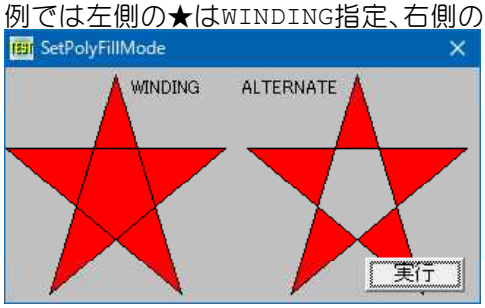
```
' 現在位置を受け取るバッファを参照で指定
Declare Function Api_MoveToEx& Lib "gdi32" Alias "MoveToEx" (ByVal hDC&, ByVal x&, ByVal
y&, ByVal lpPoint As Any)

' 現在の位置から終点までを直線で描画
Declare Function Api_LineTo& Lib "gdi32" Alias "LineTo" (ByVal hDC&, ByVal x&, ByVal y&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得。
その後、GDI 関数を使って、返されたデバイスコンテキスト内で描画を行える
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

#define AD_COUNTERCLOCKWISE 1                '反時計回り
#define AD_CLOCKWISE 2                       '時計回り

Var Shared Picture1 As Object
Picture1.Attach getDlgItem("Picture1")
Var Shared hDC As Long
Var Shared hWnd As Long

'=======================================
'=
'=======================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var x As Integer
    Var y As Integer

    hWnd = Picture1.GethWnd

    For x = 0 To Picture1.GetWidth Step 10
        Picture1.Line(x, 0) - (x, picture1.GetHeight), , 14
    Next
    For y = 0 To Picture1.GetHeight Step 10
        Picture1.Line(0, y) - (Picture1.GetWidth, y), , 14
    Next
End Sub

'=======================================
'= 反時計回り
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    hDC = Api_GetDC(hWnd)

    Ret = Api_SetArcDirection(hDC, AD_COUNTERCLOCKWISE)
    Ret = Api_MoveToEx(hDC, 110, 100, 0)
    Ret = Api_ArcTo(hDC, 10, 30, 210, 180, 210, 30, 210, 180)
    Ret = Api_LineTo(hDC, 110, 100)

    Ret = Api_ReleaseDC(hWnd, hDC)
End Sub

'=======================================
'= 時計回り
'=======================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Ret As Long

    hDC = Api_GetDC(hWnd)

    Ret = Api_SetArcDirection(hDC, AD_CLOCKWISE)
    Ret = Api_MoveToEx(hDC, 120, 100, 0)
    Ret = Api_ArcTo(hDC, 20, 30, 220, 180, 220, 30, 220, 180)
    Ret = Api_LineTo(hDC, 120, 100)
```

```
    Ret = Api_ReleaseDC(hWnd, hDC)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

多角形の塗りつぶし

多角形(★)を描画し塗り潰します。
SetPolyFillMode 多角形塗りつぶしモードを設定
GetPolyFillMode 多角形塗りつぶしモードを取得
Polygon 直線により接続された2つ以上の頂点からなっているポリゴンを引く
CreateSolidBrush ソリッドカラーで論理ブラシを作成
SelectObject 指定されたデバイスコンテキストのオブジェクトを選択
DeleteObject オブジェクトの削除
GetDC デバイスコンテキストの取得
ReleaseDC デバイスコンテキストを解放

例では左側の★はWINDING指定、右側の★はALTERNATE指定で塗り潰しています。




```
'=====================================
'= 多角形の塗り潰し
'=    (SetPolyFillMode.bas)
'=====================================
#include "Windows.bi"

Type POINTAPI
    x As Long
    y As Long
End Type

' 多角形塗りつぶしモードを設定
Declare Function Api_SetPolyFillMode& Lib "gdi32" Alias "SetPolyFillMode" (ByVal hDC&,
ByVal nPolyFillMode&)

' 多角形塗りつぶしモードを取得
Declare Function Api_GetPolyFillMode& Lib "gdi32" Alias "GetPolyFillMode" (ByVal hDC&)

' 直線により接続された2つ以上の頂点から成っているポリゴンを引く
Declare Function Api_Polygon& Lib "gdi32" Alias "Polygon" (ByVal hDC&, lpPoint As Any,
ByVal nCount&)

' ソリッドカラーで論理ブラシを作成
Declare Function Api_CreateSolidBrush& Lib "gdi32" Alias "CreateSolidBrush" (ByVal
crColor&)

' 指定されたデバイスコンテキストのオブジェクトを選択
Declare Function Api_SelectObject& Lib "gdi32" Alias "SelectObject" (ByVal hDC&, ByVal
hObject&)
```

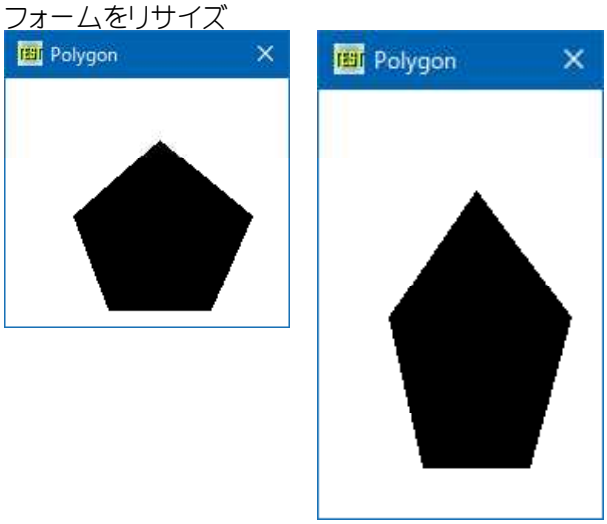
```
' ペン、ブラシ、フォント、ビットマップ、リージョン、パレットのいずれかの論理オブジェクトを削除し、そのオブジェクトに関連付けられていたすべてのシステムリソースを解放。オブジェクトを削除した後は、指定されたハンドルは無効になる
Declare Function Api_DeleteObject& Lib "gdi32" Alias "DeleteObject" (ByVal hObject&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

#define ALTERNATE 1                         '交差したリージョンは塗らない
#define WINDING 2                           '交差したリージョンも塗る
#define FLOODFILLBORDER 0                   '塗りつぶし領域がcrColorで指定された色に囲まれている範囲

Var Shared Text1 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 12
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var papi(4) As POINTAPI
    Var hDC As Long
    Var rgbColor As Long
    Var hBrush As Long
    Var i As Integer
    static StarWidth As Long

    'デバイスコンテキストを取得
    hDC = Api_GetDC(GethWnd)

    '★の幅を設定
    StarWidth = 150

    '塗り潰し色を赤に設定
    rgbColor = RGB(255, 0, 0)

    '左側の★のポイントを設定
    papi(0).x = StarWidth / 2
    papi(0).y = 5
    papi(1).x = StarWidth / 5 * 4
    papi(1).y = StarWidth + 5
    papi(2).x = 0
    papi(2).y = StarWidth / 3 + 5
    papi(3).x = StarWidth
    papi(3).y = StarWidth / 3 + 5
    papi(4).x = StarWidth / 5
    papi(4).y = StarWidth + 5

    '塗りつぶし指定
    hBrush = Api_CreateSolidBrush(rgbColor)

    'オブジェクトを選択
    Ret = Api_SelectObject(hDC, hBrush)

    'FillModeがWINDINGでなければWINDINGに設定
    If Api_GetPolyFillMode(hDC) <> WINDING Then Ret = Api_SetPolyFillMode(hDC, WINDING)

    'ポリゴン描画
    Ret = Api_Polygon(hDC, papi(0), 5)

    '右側の★のポイントを設定
    For i = 0 To 4
        papi(i).x = papi(i).x + StarWidth + 15
    Next
```

```
    'FillModeをALTERNATEに設定
    Ret = Api_SetPolyFillMode(hDC, ALTERNATE)

    '右側の★を描画
    Ret = Api_Polygon(hDC, papi(0), 5)

    Text1.ShowWindow -1

    'デバイスコンテキストを解放・ブラシの解放
    Ret = Api_ReleaseDC(GethWnd, hDC)
    Ret = Api_ReleaseDC(hBrush, hDC)
    Ret = Api_DeleteObject(hBrush)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

多角形の描画（ポリゴン）

ここでは5角形を描画し、フォームのリサイズに合わせて拡大縮小しています。
CreatePolygonRgn 多角形のリージョンを作成
Polygon 多角形の描画
FillRgn 指定されたブラシによるリージョンの描画
GetStockObject システムで定義されているのペン、ブラシ、フォント、パレット等のハンドルを取得
DeleteObject オブジェクトの削除

フォームをリサイズ



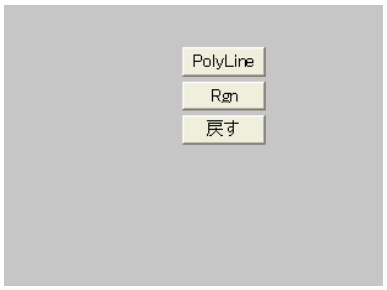

```
'=====================================
'= 多角形の描画
'=    (Polygon.bas)
'=====================================
#include "Windows.bi"

Type POINTAPI
    x   As Long
    y   As Long
End Type

' 多角形のリージョンを作成
Declare Function Api_CreatePolygonRgn& Lib "gdi32" Alias "CreatePolygonRgn" (lppt As
POINTAPI, ByVal nCount&, ByVal nPolyFillMode&)
```

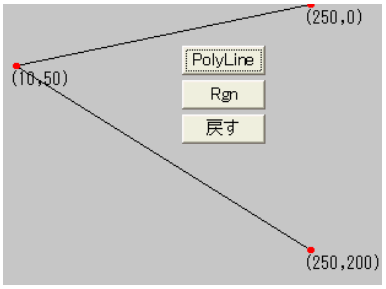
```
' 直線により接続された2つ以上の頂点から成っているポリゴンを引く
Declare Function Api_Polygon& Lib "gdi32" Alias "Polygon" (ByVal hDC&, lpPoint As Any,
ByVal nCount&)

' 指定のブラシで領域を塗りつぶす
Declare Function Api_FillRgn& Lib "gdi32" Alias "FillRgn" (ByVal hDC&, ByVal hRgn&, ByVal
hBrush&)

' ストックオブジェクトのハンドルを取得
Declare Function Api_GetStockObject& Lib "gdi32" Alias "GetStockObject" (ByVal nIndex&)

' ペン、ブラシ、フォント、ビットマップ、リージョン、パレットのいずれかの論理オブジェクトを削除し、そのオブジェクトに
関連付けられていたすべてのシステムリソースを解放。オブジェクトを削除した後は、指定されたハンドルは無効になる
Declare Function Api_DeleteObject& Lib "gdi32" Alias "DeleteObject" (ByVal hObject&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

#define ALTERNATE 1
#define WINDING 2
#define BLACKBRUSH 4

'=====================================
'=
'=====================================
Declare Sub Form_Paint edecl ()
Sub Form_Paint()
    Var poly(4) As POINTAPI
    Var NumCoords As Long
    Var hBrush As Long
    Var hRgn As Long
    Var hDC As Long
    Var Ret As Long

    Cls
    NumCoords = 5                           '5角形

    poly(0).x = GetWidth / 2
    poly(0).y = GetHeight / 5
    poly(1).x = GetWidth / 4.5
    poly(1).y = 2 * GetHeight / 4.5
    poly(2).x = GetWidth / 3
    poly(2).y = 3 * GetHeight / 4

    poly(3).x = 3 * GetWidth / 4.5
    poly(3).y = 3 * GetHeight / 4

    poly(4).x = 4 * GetWidth / 5
    poly(4).y = 2 * GetHeight / 4.5

    hDC = Api_GetDC(GethWnd)
    Ret = Api_Polygon(hDC, poly(0), NumCoords)
    hBrush = Api_GetStockObject(BLACKBRUSH)'黒のブラシ作成
    hRgn = Api_CreatePolygonRgn(poly(0), NumCoords, ALTERNATE)  '上記カラーでポリゴン作成

    If hRgn Then
        Ret = Api_FillRgn(hDC, hRgn, hBrush)
    End If

    Ret = Api_DeleteObject(hRgn)
    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Resize edecl ()
```

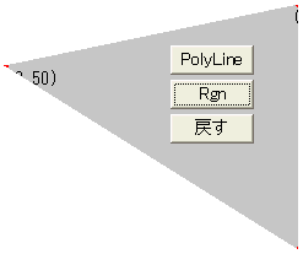
```
Sub MainForm_Resize()
    cls

    Form_Paint
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

多角形のリージョンとウィンドウ領域

CreatePolygonRgn 多角形のリージョンを作成
SetWindowRgn 指定の領域をウィンドウ領域として設定
Polyline 複数の線分からなる連続した線を描画
GetDC デバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストの解放








```
'=======================================
'= 多角形のリージョンとウィンドウ領域
'=    (CreatePolygonRgn.bas)
'=======================================
#include "Windows.bi"

Type POINTAPI
    X   As Long
    Y   As Long
End Type

' 多角形のリージョンを作成
Declare Function Api_CreatePolygonRgn& Lib "gdi32" Alias "CreatePolygonRgn" (lppt As
POINTAPI, ByVal nCount&, ByVal nPolyFillMode&)

' 指定の領域をウィンドウ領域として設定
Declare Function Api_SetWindowRgn& Lib "user32" Alias "SetWindowRgn" (ByVal hWnd&, ByVal
hRgn&, ByVal bRedraw&)

' 複数の線分からなる連続した線を描画(実行後はペンの現在位置が変更されない)
Declare Function Api_Polyline& Lib "gdi32" Alias "Polyline" (ByVal hDC&, lpPoint As
POINTAPI, ByVal nCount&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

Var Shared Button1 As Object
Var Shared Button2 As Object
Var Shared Button3 As Object
```

```
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14
Button3.Attach GetDlgItem("Button3") : Button3.SetFontSize 14

Var Shared pt(3) As POINTAPI
Var Shared hDC As Long

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    hDC = Api_GetDC(GethWnd)

    pt(0).X = 250
    pt(0).Y = 200
    pt(1).X = 10
    pt(1).Y = 50
    pt(2).X = 250
    pt(2).Y = 0
    pt(3).X = 250
    pt(3).Y = 200
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var i As Integer
    Var Ret As Long

    Ret = Api_Polyline(hDC, pt(0), 3)

    SetDrawWidth 5
    For i = 0 To 3
        Pset(pt(i).X, pt(i).Y), 5
        Symbol(pt(i).X - 4, pt(i).Y + 4), "(" & Trim$(Str$(pt(i).X)) & "," & Trim$(Str$(pt
(i).Y)) & ")", 1, 1
    Next
    SetDrawWidth 0
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var hRgn As Long
    Var Ret As Long

    hRgn = Api_CreatePolygonRgn(pt(0), 3, 1)
    Ret = Api_SetWindowRgn(GethWnd, hRgn, True)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button3_on edecl ()
Sub Button3_on()
    Var Ret As Long

    Cls
    Ret = Api_SetWindowRgn(GethWnd, 0, True)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_DblClick edecl ()
```

```
Sub MainForm_DblClick()
    Var Ret As Long

    Ret = Api_ReleaseDC(GethWnd, hDC)
    End
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

タスクバーにアイコンを追加・削除

タスクバーの右部に指定のアイコンを追加、または削除してみます。
Shell_NotifyIcon タスクトレイにアイコンを追加、変更、削除のメッセージをシステムに通知
LoadImage 画像ファイルの読み込み


```
'=====================================
'= タスクバーにアイコンを追加・削除
'=    (Shell_NotifyIcon.bas)
'=====================================
#include "Windows.bi"

Type NOTIFYICONDATA
    cbSize            As Long
    hwnd              As Long
    uID               As Long
    uFlags            As Long
    uCallbackMessage  As Long
    hIcon             As Long
    szTip             As String * 64
End Type

' タスクトレーにアイコンを追加・変更・削除のメッセージをシステムに通知
Declare Function Api_Shell_NotifyIcon& Lib "shell32" Alias "Shell_NotifyIconA" (ByVal
dwMessage&, lpData As NOTIFYICONDATA)

' 画像ファイルの読み込み
Declare Function Api_LoadImage& Lib "user32" Alias "LoadImageA" (ByVal hInst&, ByVal
lpszName$, ByVal uType&, ByVal cxDesired&, ByVal cyDesired&, ByVal fuLoad&)

#define NIF_ICON &H2                    'hIconが有効
#define NIF_MESSAGE &H1                 'uCallbackMessageが有効
#define NIF_TIP &H4                     'szTipが有効
#define NIM_ADD &H0                     'アイコンを追加
#define NIM_DELETE &H2                  'アイコンを削除
#define NIM_MODIFY &H1                  'アイコンを変更
#define WM_MOUSEMOVE &H200              'マウスが移動した
#define IMAGE_BITMAP 0                  'ビットマップ
#define IMAGE_CURSOR 2                  'カーソル
```

```
#define IMAGE_ENHMETAFILE 3                    '拡張メタファイル
#define IMAGE_ICON 1                           'アイコン
#define LR_LOADFROMFILE &H10                   '外部ファイルからロードする

Var Shared nid As NOTIFYICONDATA

Var Shared Button1 As Object
Var Shared Button2 As Object

Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14

'=====================================
'= 追加
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long
    nid.cbSize = Len(nid)
    nid.hwnd = GethWnd
    nid.uID = 1
    nid.uFlags = NIF_MESSAGE Or NIF_ICON Or NIF_TIP
    nid.uCallbackMessage = WM_MOUSEMOVE
    nid.hIcon = Api_LoadImage(GethInst, "Test5.ico", IMAGE_ICON, 0, 0, LR_LOADFROMFILE)
    nid.szTip = "これは Tooltip :-)" & Chr$(0)

    Ret = Api_Shell_NotifyIcon(NIM_ADD, nid)
End Sub

'=====================================
'= 削除
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Ret As Long

    Ret = Api_Shell_NotifyIcon(NIM_DELETE, nid)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
    Button2_on
    End
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

タスクバーの位置とサイズの取得

タスクバーの位置とサイズを取得します。
SHAppBarMessage アプリケーションバーのメッセージをシステムに送る

WindowsXP例　　Windows98例


```
'=====================================
'=  タスクバーの位置とサイズ
'=    (AppBarMsg.bas)
'=====================================
#include "Windows.bi"

Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

Type APPBARDATA
    cbSize              As Long
    hWnd                As Long
    uCallbackMessage    As Long
    uEdge               As Long
    rc                  As RECT
    lParam              As Long
End Type

' アプリケーションバーのメッセージをシステムに送る
Declare Function Api_SHAppBarMessage& Lib "Shell32" Alias "SHAppBarMessage" (ByVal
dwMessage&, pData As APPBARDATA)

#define ABM_GETTASKBARPOS &H5               'アプリケーションバーの位置とサイズを調べる

Var Shared Text1 As Object
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var AppData As APPBARDATA
    Var txt As String
    Var CrLf As String
    Var Ret As Long

    CrLf = Chr$(13,10)

    'タスクバーの位置とサイズを取得します
    Ret = Api_SHAppBarMessage(ABM_GETTASKBARPOS, AppData)

    txt =       " 縦位置(Top) : " & Format$(AppData.rc.Top,"#,###") & CrLf
    txt = txt & " 横位置(Left): " & Format$(AppData.rc.Left,"#,###") & CrLf
    txt = txt & " 幅(Width)   : " & Format$(AppData.rc.Right - AppData.rc.Left,"#,###") &
CrLf
    txt = txt & " 高さ(Height): " & Format$(AppData.rc.Bottom - AppData.rc.Top,"#,###")
    Text1.SetWindowText txt
End Sub

'=====================================
'=
'=====================================
```

```
While 1
    WaitEvent
Wend
Stop
End
```

タスクバーの設定状態を取得

SHAppBarMessage タスクバーの位置及び設定状態を取得

XGAの場合


```
'=====================================
'= タスクバーの設定状態を取得
'=    (SHAppBarMessage.bas)
'=====================================
#include "Windows.bi"

#define ABS_AUTOHIDE &H1
#define ABS_ONTOP &H2
#define ABM_GETSTATE &H4
#define ABM_GETTASKBARPOS &H5

Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

Type APPBARDATA
    cbSize      As Long
    hwnd        As Long
    uCallbackMessage As Long
    uEdge       As Long
    rc          As RECT
    lParam      As Long
End Type
```

```
Declare Function Api_SHAppBarMessage& Lib "shell32" Alias "SHAppBarMessage" (ByVal
dwMessage&, pData As APPBARDATA)

Var Shared Text1 As Object
Var Shared Check1 As Object
Var Shared Check2 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Check1.Attach GetDlgItem("Check1") : Check1.SetFontSize 14
Check2.Attach GetDlgItem("Check2") : Check2.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
        Var ABD As APPBARDATA
        Var Ret As Long

        Ret = Api_SHAppBarMessage(ABM_GETTASKBARPOS, ABD)        'タスクバーの座標を取得
        Ret = Api_SHAppBarMessage(ABM_GETSTATE, ABD)             'タスクバーの状態を取得
    If (Ret and ABS_AUTOHIDE) Then Check1.SetCheck 1 else Check1.SetCheck 0
        If (Ret and ABS_ONTOP) Then Check2.SetCheck 1 else Check2.SetCheck 0
    Text1.SetWindowText "タスクバーの座標: (" + Trim$(Str$(ABD.rc.Left)) + "," + Trim$(Str$
(ABD.rc.Top)) + ")-(" + Trim$(Str$(ABD.rc.Right)) + "," + Trim$(Str$(ABD.rc.Bottom)) +
")"
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

タスクバーの表示・非表示(ＩＩ)

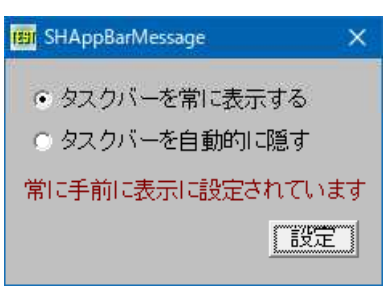
タスクバーを表示・非表示に切り替えます。
ShowWindow 指定されたウィンドウの表示状態を設定
FindWindow 指定された文字列と一致するクラス名とウィンドウ名を持つトップレベルウィンドウ

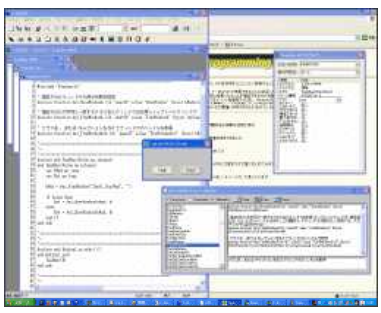

```
'=====================================
'= タスクバーの表示・非表示
'=    (TaskBar.bas)
'=====================================
#include "Windows.bi"

' 指定されたウィンドウの表示状態を設定
Declare Function Api_ShowWindow& Lib "user32" Alias "ShowWindow" (ByVal hWnd&,ByVal
nCmdShow&)

' 指定された文字列と一致するクラス名とウィンドウ名を持つトップレベルウィンドウ
Declare Function Api_FindWindow& Lib "user32" Alias "FindWindowA" (ByVal lpClassName$,
```

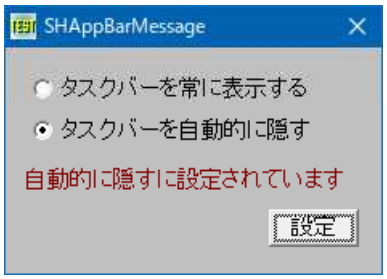
```
ByVal lpWindowName$)

Var Shared Button(1) As Object

For i = 0 To 1
    Button(i).Attach GetDlgItem("Button" & Trim$(Str$(i + 1)))
    Button(i).SetFontSize 14
Next i

'=======================================
'=
'=======================================
Declare Sub TaskBar(Value As Integer)
Sub TaskBar(Value As Integer)
    Var hWnd As Long
    Var Ret As Long

    hWnd = Api_FindWindow("Shell_TrayWnd", "")

    If Value Then
        Ret = Api_ShowWindow(hWnd, 5)
    Else
        Ret = Api_ShowWindow(hWnd, 0)
    End If
End Sub

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    TaskBar(0)
End Sub

'=======================================
'=
'=======================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    TaskBar(1)
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

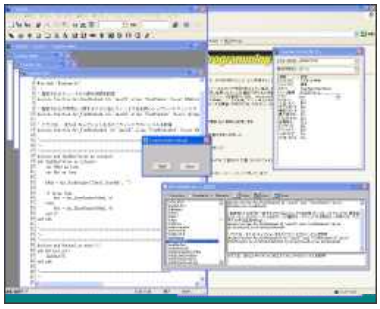
タスクバーの表示･非表示(II)

タスクバーを手前に表示･自動的に隠すを切り替えます。(WindowsXPのみ適用可)
SHAppBarMessage アプリケーションバーのメッセージをシステムに送る







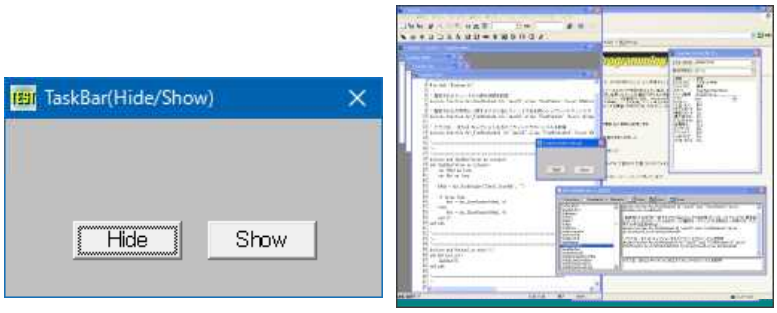


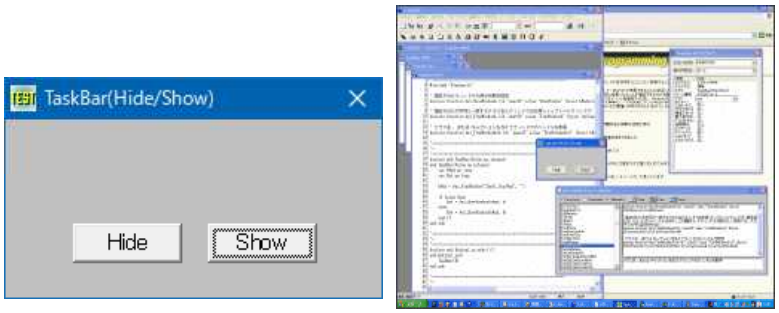
```
'=====================================
'= タスクバーの表示･非表示
'=    (SHAppBarMessage2.bas)
'=====================================
#include "Windows.bi"

#define ABM_GETSTATE &H4                    '自動的に隠す･常に手前に表示の状態を取得
#define ABM_SETSTATE &HA                    '自動的に隠すを設定
#define ABS_AUTOHIDE &H1                    '自動で隠す
#define ABS_ALWAYSONTOP &H2                 '常に手前に表示

Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

Type APPBARDATA
    cbSize      As Long
    hwnd        As Long
    uCallbackMessage As Long
    uEdge       As Long
    rc          As RECT
    lParam      As Long
End Type

' アプリケーションバーのメッセージをシステムに送る
Declare Function Api_SHAppBarMessage& Lib "Shell32" Alias "SHAppBarMessage" (ByVal
dwMessage&, pData As APPBARDATA)

Var Shared Text1 As Object
Var Shared Radio1 As Object
Var Shared Radio2 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Radio1.Attach GetDlgItem("Radio1") : Radio1.SetFontSize 14
Radio2.Attach GetDlgItem("Radio2") : Radio2.SetFontSize 14

'=====================================
'=
'=====================================
Declare Function Index bdecl () As Integer
Function Index() As Integer
    Index = val(Mid$(GetDlgRadioSelect("Radio1"), 6)) - 1
    Text1.SetWindowText "常に手前に表示に設定されています"
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var AppData As APPBARDATA
    Var Ret As Long

    If Index = 0 Then
        AppData.lParam = ABS_ALWAYSONTOP    '常に手前に表示設定
```

```
    Else
        AppData.lParam = ABS_AUTOHIDE        '自動で隠すの設定
    End If

    'タスクバー自動で隠す・手前に表示設定
    Ret = Api_SHAppBarMessage(ABM_SETSTATE, AppData)

    'タスクバーの設定状態を取得
    Ret = Api_SHAppBarMessage(ABM_GETSTATE, AppData)

    If Ret = ABS_ALWAYSONTOP Then
        Text1.SetWindowText "常に手前に表示に設定されています"
    End If

    If Ret = ABS_AUTOHIDE Then
        Text1.SetWindowText "自動的に隠すに設定されています"
    End If
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

タスクバーの表示・非表示（III）

タスクバーを表示・非表示に切り替えます。
FindWindow クラス名またはキャプションを与えてウィンドウのハンドルを取得
SetWindowPos ウィンドウのサイズ、位置、および Z オーダーを設定



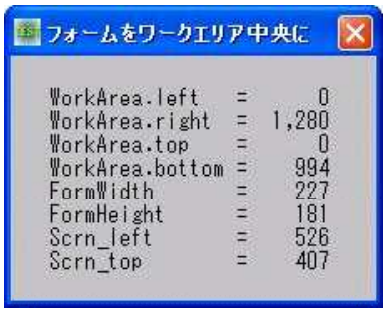


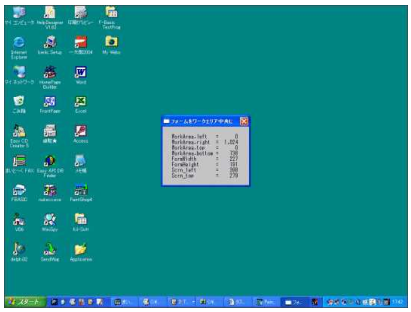

```
'=======================================
'= タスクバーの表示・非表示（III）
'=    (SetWindowPos5.bas)
'=======================================
#include "Windows.bi"

' クラス名またはキャプションを与えてウィンドウのハンドルを取得
Declare Function Api_FindWindow& Lib "user32" Alias "FindWindowA" (ByVal lpClassName$,
ByVal lpWindowName$)
```

```
' ウィンドウのサイズ、位置、および Z オーダーを設定
Declare Function Api_SetWindowPos& Lib "user32" Alias "SetWindowPos" (ByVal hWnd&, ByVal
hWndInsertAfter&, ByVal X&, ByVal Y&, ByVal CX&, ByVal CY&, ByVal uFlags&)

#define SWP_HIDEWINDOW &H80                    'ウィンドウを隠す
#define SWP_SHOWWINDOW &H40                    'ウィンドウを表示する

'=======================================
'= 非表示
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    Ret = Api_FindWindow("Shell_TrayWnd", ByVal 0)
    Ret = Api_SetWindowPos(Ret, 0, 0, 0, 0, 0, SWP_HIDEWINDOW)
End Sub

'=======================================
'= 表示
'=======================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Ret As Long

    Ret = Api_FindWindow("Shell_TrayWnd", ByVal 0)
    Ret = Api_SetWindowPos(Ret, 0, 0, 0, 0, 0, SWP_SHOWWINDOW)
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

タスクバーを考慮してフォームを画面中央に(Ⅰ)

タスクバーの高さを考慮してフォームを画面中央に表示します。
FORMは、プロパティで可視なしに設定しておき起動時表示位置を確定後表示させます。
SystemParametersInfo システムパラメータ情報の取得





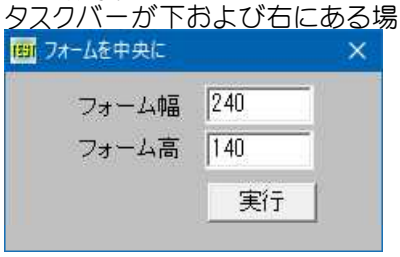
```
'=======================================
'= タスクバーを考慮してフォームを画面中央に
'= MAINFORMは、可視なしに設定しておく
'=======================================
#include "Windows.bi"

' WorkArea
Type RECT
    LEFT        As Long
    TOP         As Long
    RIGHT       As Long
```

```
    BOTTOM     As Long
End Type

' システムパラメータ情報の取得API
Declare Function API_SYSTEMPARAMETERSINFO& Lib "user32" Alias "SystemParametersInfoA"
(ByVal UACTION&, ByVal UPARAM&, ByRef LPVPARAM As Any, ByVal FUWININI&)

#define SPI_GETWORKAREA 48                    '主モニターの有効なスクリーンのサイズを取得

Var Shared MAINFORM As Object
Var Shared TEXT1 As Object
MAINFORM.ATTACH GETHWND
TEXT1.ATTACH GETDLGITEM("TEXT1")
TEXT1.SETFONTNAME "MS ゴシック"
TEXT1.SETFONTSIZE 14

'=====================================
'=
'=====================================
Declare Sub MAINFORM_START edecl ()
Sub MAINFORM_START()
    Var CRLF$ As String
    Var RES As Long
    Var WORKAREA As RECT

    CRLF$ = Chr$(13,10)

    ' ワークエリア取得
    RES = API_SYSTEMPARAMETERSINFO(SPI_GETWORKAREA, 0, WORKAREA, 0)

    SCRN_LEFT = (((WORKAREA.RIGHT - WORKAREA.LEFT) - MAINFORM.GETWIDTH) / 2) +
WORKAREA.LEFT
    SCRN_TOP = (((WORKAREA.BOTTOM - WORKAREA.TOP) - MAINFORM.GETHEIGHT) / 2) +
WORKAREA.TOP
    MAINFORM.MOVEWINDOW SCRN_LEFT, SCRN_TOP

    ' 位置決定後表示させる
    MAINFORM.SHOWWINDOW -1

    TXT$ =       "WorkArea.left   = " & Format$(WORKAREA.LEFT     ,"##,###") & CRLF$
    TXT$ = TXT$ & "WorkArea.right  = " & Format$(WORKAREA.RIGHT    ,"##,###") & CRLF$
    TXT$ = TXT$ & "WorkArea.top    = " & Format$(WORKAREA.TOP      ,"##,###") & CRLF$
    TXT$ = TXT$ & "WorkArea.bottom = " & Format$(WORKAREA.BOTTOM   ,"##,###") & CRLF$
    TXT$ = TXT$ & "FormWidth       = " & Format$(MAINFORM.GETWIDTH ,"##,###") & CRLF$
    TXT$ = TXT$ & "FormHeight      = " & Format$(MAINFORM.GETHEIGHT,"##,###") & CRLF$
    TXT$ = TXT$ & "Scrn_left       = " & Format$(SCRN_LEFT         ,"##,###") & CRLF$
    TXT$ = TXT$ & "Scrn_top        = " & Format$(SCRN_TOP          ,"##,###") & CRLF$
    TEXT1.SETWINDOWTEXT TXT$
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

タスクバーを考慮しフォームを画面中央に(II)

タスクバーを考慮しフォームをデスクトップ中央に配置します。
FindWindow クラス名またはキャプションを与えてウィンドウのハンドルを取得
GetWindowRect ウィンドウの座標をスクリーン座標系で取得
GetDesktopWindow Windows のデスクトップ ウィンドウを識別
GetClientRect ウィンドウのクライアント領域の座標を取得

画面解像度により、適当なフォームサイズを指定し、中央に配置されるか確認しています。(疑い深い••)
タスクバーが下および右にある場合のみ想定しています。

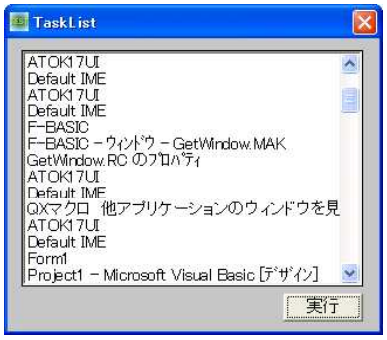



```
'=====================================
'= タスクバーを考慮しフォームを中央に(II)
'=    (MoveWindowCenter.bas)
'=====================================
#include "Windows.bi"

Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

' クラス名またはキャプションを与えてウィンドウのハンドルを取得
Declare Function Api_FindWindow& Lib "user32" Alias "FindWindowA" (ByVal lpClassName$,
ByVal lpWindowName$)

' ウィンドウの座標をスクリーン座標系で取得
Declare Function Api_GetWindowRect& Lib "user32" Alias "GetWindowRect" (ByVal hWnd&,
lpRect As RECT)

' Windows のデスクトップ ウィンドウを識別。返されるポインタは、一時的なポインタ。後で使用するために保存しておくことはできない
Declare Function Api_GetDesktopWindow& Lib "user32" Alias "GetDesktopWindow" ()

' ウィンドウのクライアント領域の座標を取得
Declare Function Api_GetClientRect& Lib "user32" Alias "GetClientRect" (ByVal hWnd&,
lpRect As RECT)

Var Shared Text(1) As Object
Var Shared Edit(1) As Object
Var Shared Button1 As Object

For i = 0 To 1
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1))) : Text(i).SetFontSize 14
    Edit(i).Attach GetDlgItem("Edit" & Trim$(Str$(i + 1))) : Edit(i).SetFontSize 14
Next i
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Rect_Client As RECT
    Var Rect_TaskBar As RECT
    Var fWidth As Long
    Var fHeight As Long
    Var vLeft As Long
    Var vTop As Long
    Var hTask As Long
    Var Ret As Long

    fWidth = Val(Edit(0).GetWindowText)
    fHeight = Val(Edit(1).GetWindowText)
    If fWidth < 240 Or fHeight < 140 Then
        fWidth = 240
        fHeight = 140
    End If
```

```
    SetWindowSize fWidth, fHeight

    Ret = Api_GetClientRect(Api_GetDesktopWindow(), Rect_Client)
    hTask = Api_FindWindow("Shell_TrayWnd", Chr$(0))

    If hTask <> 0 Then
        Ret = Api_GetWindowRect(hTask, Rect_TaskBar)
        If (Rect_TaskBar.Right - Rect_TaskBar.Left) > (Rect_TaskBar.Bottom -
Rect_TaskBar.Top) Then
            If Rect_TaskBar.Top <= 0 Then
                Rect_Client.Top = Rect_Client.Top + Rect_TaskBar.Bottom
            Else
                Rect_Client.Bottom = Rect_Client.Bottom - (Rect_TaskBar.Bottom -
Rect_TaskBar.Top)
            End If
        Else
            If Rect_TaskBar.Left <= 0 Then
                Rect_Client.Left = Rect_Client.Left + Rect_TaskBar.Right
            Else
                Rect_Client.Right = Rect_Client.Right - (Rect_TaskBar.Right -
Rect_TaskBar.Left)
            End If
        End If
    End If

    vLeft = (Rect_Client.Right - Rect_Client.Left - GetWidth) / 2
    vTop = (Rect_Client.Bottom - Rect_Client.Top - GetHeight) / 2
    MoveWindow vLeft, vTop
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

タスクリストを取得

タスクリストを取得し表示します。
GetWindow 指定されたウィンドウと指定された関係にあるウィンドウのハンドルを取得
GetParent 指定されたウィンドウの親ウィンドウまたはオーナーウィンドウのハンドルを取得
GetWindowTextLength ウィンドウのタイトル文字数を取得
GetWindowText ウインドウのタイトル文字列を取得




```
'=======================================
'= タスクリストの表示
'=    (TaskList.bas)
'=======================================
#include "Windows.bi"
```

```
' 指定されたウィンドウと指定された関係にあるウィンドウのハンドルを取得
Declare Function Api_GetWindow& Lib "user32" Alias "GetWindow" (ByVal hWnd&, ByVal wCmd&)

' 指定されたウィンドウの親ウィンドウまたはオーナーウィンドウのハンドルを取得
Declare Function Api_GetParent& Lib "user32" Alias "GetParent" (ByVal hWnd&)

' ウィンドウのタイトル文字数を取得
Declare Function Api_GetWindowTextLength& Lib "user32" Alias "GetWindowTextLengthA"
(ByVal hWnd&)

' ウインドウのタイトル文字列を取得
Declare Function Api_GetWindowText& Lib "user32" Alias "GetWindowTextA" (ByVal hWnd&,
ByVal lpString$, ByVal cch&)

#define GW_HWNDFIRST 0                          '最初のウィンドウ
#define GW_HWNDNext 2                           '次のウィンドウ

Var Shared List1 As Object
Var Shared Button1 As Object

List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub TaskListGet()
Sub TaskListGet()
    Var CurrWnd As Long
    Var Length As Long
    Var TaskName As String
    Var Parent As Long

    List1.ResetContent
    CurrWnd = Api_GetWindow(GethWnd, GW_HWNDFIRST)
    While CurrWnd <> 0
        Parent = Api_GetParent(CurrWnd)
        Length = Api_GetWindowTextLength(CurrWnd)
        TaskName = space$(Length + 1)
        Length = Api_GetWindowText(CurrWnd, TaskName, Length + 1)
        TaskName = Left$(TaskName, Len(TaskName) - 1)
        If Length <> 0 Then
            If TaskName <> GetWIndowText Then
                List1.AddString TaskName
            End If
        End If
        CurrWnd = Api_GetWindow(CurrWnd, GW_HWNDNext)
        CallEvent
    Wend
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    TaskListGet
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

他端末のレジストリ情報を取得

RegConnectRegistry ほかのコンピュータ上の定義済みレジストリハンドルとの接続を確立
RegCloseKey レジストリのハンドルを解放
RegOpenKeyEx レジストリのキーのハンドルを確保
RegQueryValueEx レジストリの値を取得

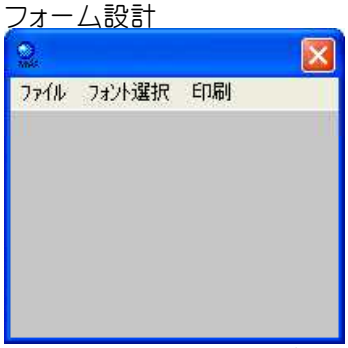

```
'=======================================
'= 他端末のレジストリ情報を取得
'=    (RegConnectRegistry2.bas)
'=======================================
#include "Windows.bi"

#define HKEY_CLASSES_ROOT -2147483648      '拡張子に関する情報や、それらとアプリケーションとの関連づけに関する情報
#define HKEY_CURRENT_USER -2147483647      '現在Windowsにログインしているユーザーの情報
#define HKEY_LOCAL_MACHINE -2147483646     'PCを利用するユーザーに共通の設定情報
#define HKEY_USERS -2147483645             'Windowsを利用するユーザー個別の情報
#define KEY_QUERY_VALUE &H1                'サブキーデータを問い合わせるためのアクセス権
#define KEY_SET_VALUE &H2                  'サブキーデータの設定を許可
#define KEY_ALL_ACCESS &H3F                'レジストリに対するすべての権限を許可
#define REG_SZ 1                           'ヌル終端文字列
#define ERROR_SUCCESS &H0                  '正常終了の戻り値を示す

' ほかのコンピュータ上の定義済みレジストリハンドルとの接続を確立
Declare Function Api_RegConnectRegistry& Lib "advapi32" Alias "RegConnectRegistryA"
(ByVal lpMachineName$, ByVal hKey&, phkResult&)

' レジストリのハンドルを解放
Declare Function Api_RegCloseKey& Lib "advapi32" Alias "RegCloseKey" (ByVal hKey&)

' レジストリのキーのハンドルを確保
Declare Function Api_RegOpenKeyEx& Lib "advapi32" Alias "RegOpenKeyExA" (ByVal hKey&,
ByVal lpSubKey$, ByVal ulOptions&, ByVal samDesired&, phkResult&)

' レジストリの値を取得
Declare Function Api_RegQueryValueEx& Lib "advapi32" Alias "RegQueryValueExA" (ByVal
hKey&, ByVal lpvName$, ByVal lpReserved&, lpType&, ByVal lpData$, lpcbData&)

Var Shared Edit1 As Object
Var Shared Edit2 As Object
Var Shared Text1 As Object
Var Shared Button1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Edit2.Attach GetDlgItem("Edit2") : Edit2.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var Shared hRemoteReg As Long

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hKey As Long
    Var sValue As String
    Var RemoteName As String
```

```
    Var KeyName As String
    Var Ret As Long

    RemoteName = Edit1.GetWindowText

    'リモート端末の接続(空文字の場合はローカルコンピューター)
    Ret = Api_RegConnectRegistry(RemoteName, HKEY_LOCAL_MACHINE, hRemoteReg)

    If (Ret = ERROR_SUCCESS) Then
        Text1.SetWindowtext "接続されました！"
    Else
        Text1.SetWindowtext "エラー！"

        If hRemoteReg <> 0 Then
            Ret = Api_RegCloseKey(hRemoteReg)
        End If
        Exit Sub
    End If

    Wait 50

    KeyName = Edit2.GetWindowText

    Ret = Api_RegOpenKeyEx(hRemoteReg, "HARDWARE¥DESCRIPTION¥System", 0,
KEY_QUERY_VALUE, hKey)

    If Ret <> ERROR_SUCCESS Then
        Text1.SetWindowtext "オープンできません！"
    Else
        sValue = String$(255, " ")
        Ret = Api_RegQueryValueEx(hKey, KeyName, 0&, REG_SZ, sValue, 255)

        If Ret <> ERROR_SUCCESS Then
            Text1.SetWindowtext "取得できません！"
        Else
            Text1.SetWindowtext sValue
        End If

        Ret = Api_RegCloseKey(hKey)

        If Ret <> ERROR_SUCCESS Then
            Text1.SetWindowtext "クローズできません！"
        End If
    End If
End Sub

'=======================================
'=
'=======================================
Declare Sub Edit1_Change edecl ()
Sub Edit1_Change()
    Text1.SetWindowText ""
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

縦書き例:はがき印刷

縦書き表示例。ついでにはがきを印刷してみます。フォントの変更も可能です。
初期状態はMS ゴシックに設定されています。フォント変更時@の付いたフォントおよびプロポーショナルフォントは選択

しないでください。
フォーム設計



宛先の郵便番号はMS ゴシック固定です。宛先郵便番号、宛名、差出人名前および電話番号の部分は均等割付をしています。
スタンプ(50円)はSTAMP.BMPをロードしていますが、ない場合は50円切手と表示されます。
右:文字サイズはプログラム上で指定していますので無効です。
if choosefont(ffont, 0, 0, 0) then の場合

左:if choosefont(ffont, rgbcolor, -1, 0) then の場合
右:if choosefont(ffont, rgbcolor, -1, -1) then の場合


```
'=======================================
'=　縦書き一例･はがき印刷
'=======================================
#include "Windows.bi"

Var Shared MAINFORM As Object
Var Shared PRT As Object
Var Shared BITMAP As Object

Var Shared F_NAME$ As String                'フォント名
Var Shared F_POINT As single                '文字ポイント
Var Shared F_SPACE As single                '文字間隔(通常はフォントサイズ)
Var Shared F_FRAME As single                '均等割付時の文字枠長
Var Shared FFONT As FONT                    'フォント構造体
Var Shared LM As Integer                    '文字列長さ
Var Shared PRM As PRINTPARAM
Var Shared ATENA$(3) As String
Var Shared SEnd$(4) As String
Var Shared TXT$ As String
Var Shared XPOS As Integer
Var Shared YPOS As Integer
Var Shared CARDW As Integer
Var Shared CARDH As Integer
```

```
Var Shared LX As Integer
Var Shared LY As Integer

def fnWD(X) = F_POINT/72*254                    '文字ポイントを1/10mm(MAPMODE=2)に合わす

PRINTERObject PRT
BITMAPObject BITMAP

Declare Sub MAINFORM_START edecl ()
Declare Sub POSTCARD_DRAW edecl ()
Declare Sub ATENA_DRAW edecl ()
Declare Sub TATEGAKI_DRAW edecl ()
Declare Sub YOKOGAKI_DRAW edecl ()
Declare Sub MNUFONTSEL_ON edecl ()
Declare Sub MNUEXIT_ON edecl ()
Declare Sub MAINFORM_QUERYCLOSE edecl (CANCEL%,ByVal MODE%)


'=======================================
'=
'=======================================
Sub MAINFORM_START()
    If PREVINSTANCE Then
        RES = MESSAGEBOX(GETWINDOWTEXT, "このアプリケーションは実行中です！", 0, 0) : End
    End If

    ATENA$(0) = "佐々木　小次郎"
    ATENA$(1) = "064-0923"
    ATENA$(2) = "北海道札幌市中央区南12条西34丁目5-6"
    ATENA$(3) = "巌流マンション803"
    SEnd$(0) = "阪神　虎之助"
    SEnd$(1) = "654-0103"
    SEnd$(2) = "兵庫県神戸市須磨区黒山台5丁目67-8"
    SEnd$(3) = "タイガーハイツ603"
    SEnd$(4) = "電話・FAX078-789-0123"

    SETWINDOWTEXT "縦書例:はがき印刷"
    CARDW = 1000
    CARDH = 1480

    SETMAPMODE 2
    SETWINDOWSIZE CARDW+60, CARDH+170
    MOVEWINDOW 50, 50
    SHOWWINDOW -1

    F_NAME$ = "MS ゴシック"
    FFONT.FFNAME = F_NAME$
    POSTCARD_DRAW
    ATENA_DRAW
End Sub

'=======================================
'=
'=======================================
Sub POSTCARD_DRAW()
    on error goto *ER_TRAP

    LX = 20
    LY = 20
    line(LX, LY)-(LX+CARDW, LY+CARDH),,15,bf
    For X = 435 To 575 step 70
        line(X+LX, 120+LY)-(X+60+LX, 200+LY),,5,b
        line(X+LX+2, 120+LY+2)-(X+60+LX-2, 200+LY-2),,5,b
    Next
    For X = 650 To 860 step 70
        line(X+LX, 120+LY)-(X+60+LX, 200+LY),,5,b
    Next
    line(635+LX, 160+LY)-(650+LX, 160+LY),,5

    BITMAP.LOADFILE "STAMP.BMP"
```

```
    STRETCHBITMAP BITMAP, 90+LX, 110+LY, 180, 250
    BITMAP.DELETEObject
    Exit Sub

*ER_TRAP 'STAMP.BMPがない場合
    line(90+LX, 110+LY)-(270+LX, 360+LY),preset,8,bf
    SETFONTNAME "MS ゴシック"
    SETFONTSIZE 12
    SETFONTBOLD -1
    symbol(135+LX, 180+LY),"50円",1,1,15
    symbol(135+LX, 230+LY),"切手",1,1,15
    resume Next
End Sub

'=====================================
'=  F_FRAME:均等割付の長さ
'=====================================
Sub ATENA_DRAW()

    ' 郵便番号(3桁部)
    F_POINT = 12
    TXT$ = akcnv$(Left$(ATENA$(1),instr(ATENA$(1),"-") - 1))
    LM = KLen(TXT$)
    F_FRAME = 180
    F_SPACE = fnWD(F_POINT) + ((F_FRAME - LM * fnWD(F_POINT)) / (LM-1))
    XPOS = 450
    YPOS = 145
    YOKOGAKI_DRAW

    ' 郵便番号(4桁部)
    TXT$ = akcnv$(Mid$(ATENA$(1),instr(ATENA$(1),"-") + 1))
    LM = KLen(TXT$)
    F_FRAME = 240
    F_SPACE = fnWD(F_POINT) + ((F_FRAME - LM * fnWD(F_POINT)) / (LM-1))
    XPOS = 665
    YPOS = 145
    YOKOGAKI_DRAW

    ' 住所1
    F_POINT = 12
    TXT$ = akcnv$(ATENA$(2))
    LM = KLen(TXT$)
    F_SPACE = fnWD(F_POINT)
    XPOS = 880
    YPOS = 270
    TATEGAKI_DRAW

    ' 住所2
    TXT$ = akcnv$(ATENA$(3))
    LM = KLen(TXT$)
    F_SPACE = fnWD(F_POINT)
    XPOS = 800
    YPOS = 320
    TATEGAKI_DRAW

    ' 名前 & 敬称
    TXT$ = akcnv$(ATENA$(0))+" 様"
    If LM < 15 Then F_POINT = 16 Else F_POINT = 14
    F_FRAME = 840
    F_SPACE = fnWD(F_POINT) + ((F_FRAME - LM * fnWD(F_POINT)) / (LM-1))
    XPOS = 620
    YPOS = 350
    TATEGAKI_DRAW

    ' 差出人郵便番号
    F_POINT = 8
    TXT$ = "〒" & akcnv$(SEnd$(1))
    LM = KLen(TXT$)
    F_SPACE = fnWD(F_POINT)
    XPOS = 280
```

```
    YPOS = 650
    TATEGAKI_DRAW

    ' 住所1
    F_POINT = 9
    TXT$ = akcnv$(SEnd$(2))
    LM = KLen(TXT$)
    F_SPACE = fnWD(F_POINT)
    XPOS = 240
    YPOS = 700
    TATEGAKI_DRAW

    ' 住所2
    TXT$ = akcnv$(SEnd$(3))
    LM = KLen(TXT$)
    F_SPACE = fnWD(F_POINT)
    XPOS = 200
    YPOS = 750
    TATEGAKI_DRAW

    ' 名前
    TXT$ = akcnv$(SEnd$(0))
    LM = KLen(TXT$)
    F_POINT = 11
    F_FRAME = 500
    F_SPACE = fnWD(F_POINT) + ((F_FRAME - LM * fnWD(F_POINT)) / (LM-1))
    XPOS = 160
    YPOS = 820
    TATEGAKI_DRAW

    ' 電話･FAX
    TXT$ = akcnv$(SEnd$(4))
    LM = KLen(TXT$)
    F_POINT = 8
    F_FRAME = 480
    F_SPACE = fnWD(F_POINT) + ((F_FRAME - LM * fnWD(F_POINT)) / (LM-1))
    XPOS = 110
    YPOS = 840
    TATEGAKI_DRAW
End Sub

'=====================================
'= 縦書き表示
'=====================================
Sub TATEGAKI_DRAW()
    If LM = 0 Then Exit Sub
    FFONT.FFNAME = "@"+FFONT.FFNAME
    SETFONT FFONT
    SETFONTSIZE F_POINT
    For I = 1 To LM
        ZZZ$ = kMid$(TXT$, I, 1)
        If ZZZ$ = "ー" Then
            ZZZ$ = "一"
        Else If ZZZ$ = "1" Then
            ZZZ$ = "一"
        Else If ZZZ$ = "2" Then
            ZZZ$ = "二"
        Else If ZZZ$ = "3" Then
            ZZZ$ = "三"
        Else If ZZZ$ = "4" Then
            ZZZ$ = "四"
        Else If ZZZ$ = "5" Then
            ZZZ$ = "五"
        Else If ZZZ$ = "6" Then
            ZZZ$ = "六"
        Else If ZZZ$ = "7" Then
            ZZZ$ = "七"
        Else If ZZZ$ = "8" Then
            ZZZ$ = "八"
        Else If ZZZ$ = "9" Then
```

```
            ZZZ$ = "九"
        End If
        symbol(XPOS+LX, YPOS+LY), ZZZ$, 1, 1, , 3
        YPOS = YPOS + F_SPACE
    Next
    FFONT.FFNAME = Mid$(FFONT.FFNAME, 2)
End Sub

'=====================================
'= 横書き表示
'=====================================
Sub YOKOGAKI_DRAW()
    If LM = 0 Then Exit Sub
    SETFONTNAME "MS ゴシック"
    SETFONTSIZE F_POINT
    For I = 1 To LM
        ZZZ$ = kMid$(TXT$, I, 1)
        symbol(XPOS+LX, YPOS+LY), ZZZ$, 1, 1
        XPOS = XPOS + F_SPACE
    Next
End Sub

'=====================================
'=  EPSON PX-V700:LX=-25/LY=-15
'=====================================
Declare Sub MNUPRINT_ON edecl ()
Sub MNUPRINT_ON()
    LX = -25                                    '左余白調整
    LY = -15                                    '上余白調整
    PRT.SETUPPRINTERMODE "SETUPPRINTER:", 11, 1 '
    If PRT.PRINTDLG(PRM) <> 0 Then
        PRT.SETMAPMODE 2
        PRT.STARTDOC "Sample"
        PRT.STARTPAGE

        PRT.ATENA_DRAW

        PRT.ENDPAGE
        PRT.ENDDOC
        PRT.CLOSEPRINTER
    End If
End Sub

'=====================================
'= フォント指定
'=====================================
Sub MNUFONTSEL_ON()
    FFONT.FFNAME = F_NAME$
    If CHOOSEFONT(FFONT,0,0,0) Then
        F_NAME$ = FFONT.FFNAME
    End If
    FFONT.FFNAME = F_NAME$
    POSTCARD_DRAW
    ATENA_DRAW
End Sub

'=====================================
'= 終了処理
'=====================================
Sub MAINFORM_QUERYCLOSE(CANCEL%,ByVal MODE%)
    CANCEL% = MESSAGEBOX(GETWINDOWTEXT, "終了しますか？", 1, 1 )
    If CANCEL% = 0 Then End
End Sub

Sub MNUEXIT_ON()
    RES = MESSAGEBOX(GETWINDOWTEXT, "終了しますか？", 1, 1 )
    If RES = 0 Then End
End Sub
```

```
'====================================
'=
'====================================
While 1
    WaitEvent
Wend
Stop
End
```

タブコントロールの選択インデックスを取得

GetSysColor システムの背景色を取得
CreateWindowEx ウィンドウ(コントロール)を作成
GetStockObject ストックオブジェクトのハンドルを取得
DestroyWindow CreateWindowExの解放
InitCommonControls コモンコントロールライブラリからコモンコントロールのウィンドウクラスを登録して初期化
SendMessage ウィンドウにメッセージを送信
SendMessageByLong ウィンドウにメッセージを送信
TCM_GETCURSEL(&H130B) 選択されているタブインデックス取得


```
'====================================
'= タブコントロールの選択インデックスを取得
'=    (CreateWindowEx4_7.bas)
'====================================
#include "Windows.bi"

#define COLOR_BTNFACE 15                    '3Dオブジェクトの表面色
#define DEFAULT_GUI_FONT 17                 'ユーザーインターフェイス用のデフォルトフォント
#define IDM_TAB1 &H100                      'TABコントロールのID
#define TCIF_TEXT &H1                       'pdzTextにデータが含まれる
#define TCM_INSERTITEMA &H1307              'タブ項目を追加
#define TCM_SETCURSEL &H130C                'タブを選択
#define TCM_GETCURSEL &H130B                '選択されているタブインデックス取得
#define WC_TABCONTROL "SysTabControl32"     'タブコントロール
#define WM_SETFONT &H30                     '論理フォントを設定する
#define WS_CHILD &H40000000                 '親ウィンドウを持つコントロール(子ウィンドウ)を作成する
#define WS_CLIPSIBLINGS &H4000000           '兄弟関係にある子ウィンドウをクリップする
#define WS_VISIBLE &H10000000               '可視状態のウィンドウを作成する
#define TCS_HOTTRACK &H40                   'マウスカーソルの下のタブを強調表示
#define TCS_TOOLTIPS &H4000                 'ツールヒントコントロールが作成されTTN_NEEDTEXTを発行

Type TCITEM
    mask            As Long
    dwState         As Long
    dwStateMask     As Long
    pszText         As Long
    cchTextMax      As Long
    iImage          As Long
    lParam          As Long
End Type

' システムの背景色を取得
Declare Function Api_GetSysColor& Lib "user32" Alias "GetSysColor" (ByVal nIndex&)

' ウィンドウ(コントロール)を作成
Declare Function Api_CreateWindowEx& Lib "user32" Alias "CreateWindowExA" (ByVal
ExStyle&, ByVal ClassName$, ByVal WinName$, ByVal Style&, ByVal x&, ByVal y&, ByVal
nWidth&, ByVal nHeight&, ByVal Parent&, ByVal Menu&, ByVal Instance&, Param&)
```

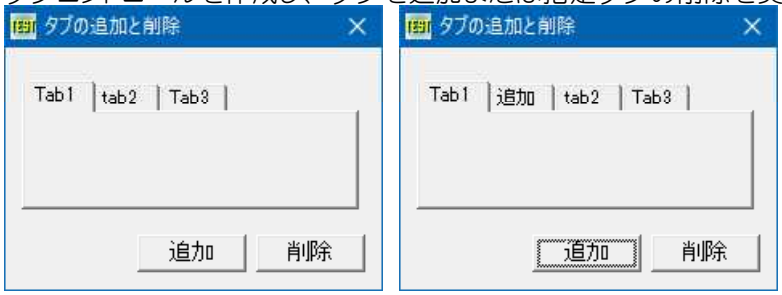
```
' ストックオブジェクトのハンドルを取得
Declare Function Api_GetStockObject& Lib "gdi32" Alias "GetStockObject" (ByVal nIndex&)

' CreateWindowExの解放
Declare Function Api_DestroyWindow& Lib "user32" Alias "DestroyWindow" (ByVal hWnd&)

' コモンコントロールライブラリからコモンコントロールのウィンドウクラスを登録して初期化
Declare Sub Api_InitCommonControls Lib "comctl32" Alias "InitCommonControls" ()

' ウィンドウにメッセージを送信
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

' ウィンドウにメッセージを送信
Declare Function Api_SendMessageByLong& Lib "user32" Alias "SendMessageA" (ByVal hWnd&,
ByVal wMsg&, ByVal wParam&, ByVal lParam&)

Var Shared Text1 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var Shared hTabs As Long

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var rgbColor As Long
    Var tci As TCITEM
    Var index As Long
    Var Ret As Long

    'Buttonの表面色を取得(F0F0F0)
    rgbColor = Api_GetSysColor(COLOR_BTNFACE)

    'MainFormを取得色で塗り
    SetBackColor rgbColor

    '画面を消去
    Cls

    'MainFormを表示
    ShowWindow -1

    Api_InitCommonControls

    hTabs = Api_CreateWindowEx(0, WC_TABCONTROL, "", WS_CHILD Or WS_CLIPSIBLINGS Or
WS_VISIBLE Or TCS_HOTTRACK Or TCS_TOOLTIPS, 10, 10, 214, 60, GethWnd, IDM_TAB1, GethInst,
ByVal 0)

    If hTabs <> 0 Then
        Ret = Api_SendMessageByLong(hTabs, WM_SETFONT,
Api_GetStockObject(DEFAULT_GUI_FONT), 0)

        tci.mask = TCIF_TEXT
        tci.pszText = StrAdr("Tab1" & Chr$(0))
        Ret = Api_SendMessage(hTabs, TCM_INSERTITEMA, 0, tci)

        tci.mask = TCIF_TEXT
        tci.pszText = StrAdr("tab2" & Chr$(0))
        Ret = Api_SendMessage(hTabs, TCM_INSERTITEMA, 1, tci)

        tci.mask = TCIF_TEXT
        tci.pszText = StrAdr("Tab3" & Chr$(0))
        Ret = Api_SendMessage(hTabs, TCM_INSERTITEMA, 2, tci)

        tci.mask = TCIF_TEXT
```

```
        tci.pszText = StrAdr("Tab4" & Chr$(0))
        Ret = Api_SendMessage(hTabs, TCM_INSERTITEMA, 3, tci)
    End If
    Ret = Api_SendMessage(hTabs, TCM_SETCURSEL, index, ByVal 0)

End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var CurSel As Long

    '選択タブインデックスを取得
    CurSel = Api_SendMessage(hTabs, TCM_GETCURSEL, 0, ByVal 0)

    '結果を表示
    Text1.SetWindowtext "Tab" & Str$(CurSel + 1) & " が選択されました"
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
    Var Ret As Long

    Ret = Api_DestroyWindow(hTabs)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

タブコントロールのタブの追加と削除

GetSysColor システムの背景色を取得
CreateWindowEx ウィンドウ(コントロール)を作成
GetStockObject ストックオブジェクトのハンドルを取得
DestroyWindow CreateWindowExの解放
InitCommonControls コモンコントロールライブラリからコモンコントロールのウィンドウクラスを登録して初期化
SendMessage ウィンドウにメッセージを送信
SendMessageByLong ウィンドウにメッセージを送信

タブコントロールを作成し、タブを追加または指定タブの削除を実行します。




```
'=====================================
'= タブコントロールのタブの追加と削除
'=    (CreateWindowEx4_4.bas)
'=====================================
#include "Windows.bi"
```

```
Type INITCOMMONCONTROLSEX
    dwSize      As Long
    dwICC       As Long
End Type

Type TCITEM
    mask                As Long
    dwState             As Long
    dwStateMask         As Long
    pszText             As Long
    cchTextMax          As Long
    iImage              As Long
    lParam              As Long
End Type

#define COLOR_BTNFACE 15                    '3Dオブジェクトの表面色
#define DEFAULT_GUI_FONT 17                 'ユーザーインターフェイス用のデフォルトフォント
#define ICC_TAB_CLASSES &H8                 'タブコントロール､ツールチップ
#define IDM_TAB1 &H100                      'TABコントロールのID
#define TCIF_TEXT &H1                       'pdzTextにデータが含まれる
#define TCM_DELETEITEM &H1308               'タブの削除
#define TCM_INSERTITEMA &H1307              'タブ項目を追加
#define TCM_GETCURSEL &H130B                '選択されているタブインデックス取得
#define TCM_SETCURSEL &H130C                'タブを選択
#define WC_TABCONTROL "SysTabControl32"     'タブコントロール
#define WM_SETFONT &H30                     '論理フォントを設定する
#define WS_CHILD &H40000000                 '親ウィンドウを持つコントロール(子ウィンドウ)を作成する
#define WS_CLIPSIBLINGS &H4000000           '兄弟関係にある子ウィンドウをクリップする
#define WS_VISIBLE &H10000000               '可視状態のウィンドウを作成する
#define TCS_HOTTRACK &H40                   'マウスカーソルの下のタブを強調表示
#define TCS_TOOLTIPS &H4000                 'ツールヒントコントロールが作成されTTN_NEEDTEXTを発行

' システムの背景色を取得
Declare Function Api_GetSysColor& Lib "user32" Alias "GetSysColor" (ByVal nIndex&)

' ウィンドウ(コントロール)を作成
Declare Function Api_CreateWindowEx& Lib "user32" Alias "CreateWindowExA" (ByVal
ExStyle&, ByVal ClassName$, ByVal WinName$, ByVal Style&, ByVal x&, ByVal y&, ByVal
nWidth&, ByVal nHeight&, ByVal Parent&, ByVal Menu&, ByVal Instance&, Param&)

' ストックオブジェクトのハンドルを取得
Declare Function Api_GetStockObject& Lib "gdi32" Alias "GetStockObject" (ByVal nIndex&)

' CreateWindowExの解放
Declare Function Api_DestroyWindow& Lib "user32" Alias "DestroyWindow" (ByVal hWnd&)

' コモンコントロールライブラリからコモンコントロールのウィンドウクラスを登録して初期化
Declare Sub Api_InitCommonControls Lib "comctl32" Alias "InitCommonControls" ()

' ウィンドウにメッセージを送信
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

' ウィンドウにメッセージを送信
Declare Function Api_SendMessageByLong& Lib "user32" Alias "SendMessageA" (ByVal hWnd&,
ByVal wMsg&, ByVal wParam&, ByVal lParam&)

Var Shared Button1 As Object
Var Shared Button2 As Object

Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14

Var Shared hTabs As Long
Var SHared tci As TCITEM

'=====================================
'=
'=====================================
```

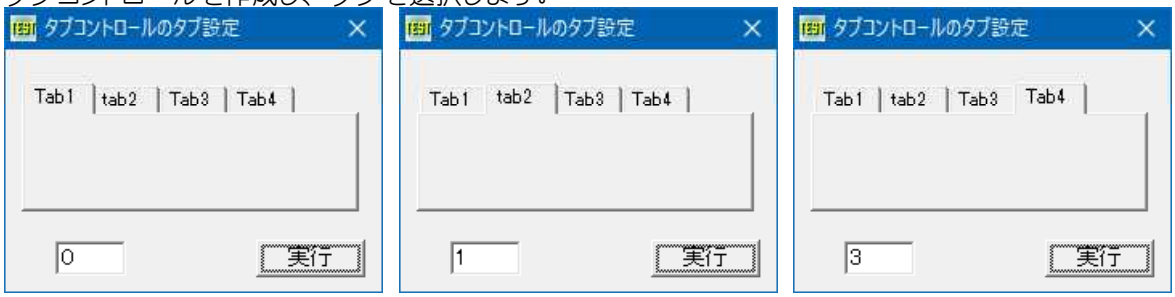
```
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var rgbColor As Long
    Var icc As INITCOMMONCONTROLSEX
    Var index As Long
    Var Ret As Long

    'Buttonの表面色を取得(EDE9EC)
    rgbColor = Api_GetSysColor(COLOR_BTNFACE)

    'MainFormを取得色で塗り
    SetBackColor rgbColor

    '画面を消去
    Cls

    'MainFormを表示
    ShowWindow -1

    icc.dwSize = Len(icc)
    icc.dwICC = ICC_TAB_CLASSES
    Api_InitCommonControls

    Ret = Api_DestroyWindow(hTabs)

    hTabs = Api_CreateWindowEx(0, WC_TABCONTROL, "", WS_CHILD Or WS_CLIPSIBLINGS Or
WS_VISIBLE Or TCS_HOTTRACK Or TCS_TOOLTIPS, 10, 20, 210, 80, GethWnd, IDM_TAB1, GethInst,
ByVal 0)

    If hTabs <> 0 Then
        Ret = Api_SendMessageByLong(hTabs, WM_SETFONT,
Api_GetStockObject(DEFAULT_GUI_FONT), 0)

        tci.mask = TCIF_TEXT
        tci.pszText = StrAdr("Tab1" & Chr$(0))
        Ret = Api_SendMessage(hTabs, TCM_INSERTITEMA, 0, tci)

        tci.mask = TCIF_TEXT
        tci.pszText = StrAdr("tab2" & Chr$(0))
        Ret = Api_SendMessage(hTabs, TCM_INSERTITEMA, 1, tci)

        tci.mask = TCIF_TEXT
        tci.pszText = StrAdr("Tab3" & Chr$(0))
        Ret = Api_SendMessage(hTabs, TCM_INSERTITEMA, 2, tci)
    End If
End Sub

'=====================================
'= タブを追加
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Insertcindex As Long
    Var InsertStr As String
    Var Ret As Long

    '設定するインデックスを指定
    Insertcindex = 1

    '設定する文字列を指定
    InsertStr = "追加"

    '設定する項目情報を構造体に設定
    tci.mask = TCIF_TEXT
    tci.pszText = StrAdr(InsertStr)
    tci.cchTextMax = Len(InsertStr)

    'タブストリップの項目を追加
    Ret = Api_SendMessage(hTabs, TCM_INSERTITEMA, Insertcindex, tci)
End Sub
```

```
'=====================================
'= 指定したタブを削除
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var SelIndex As Long
    Var DelIndex As Long

    '選択タブインデックスを取得
    SelIndex = Api_SendMessage(hTabs, TCM_GETCURSEL, 0, ByVal 0)

    '指定したタブを削除
    DelIndex = Api_SendMessage(hTabs, TCM_DELETEITEM, SelIndex, ByVal 0)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
    Var Ret As Long

    Ret = Api_DestroyWindow(hTabs)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

タブコントロールのタブを選択

GetSysColor システムの背景色を取得
CreateWindowEx ウィンドウ(コントロール)を作成
GetStockObject ストックオブジェクトのハンドルを取得
DestroyWindow CreateWindowExの解放
InitCommonControls コモンコントロールライブラリからコモンコントロールのウィンドウクラスを登録して初期化
SendMessage ウィンドウにメッセージを送信
SendMessageByLong ウィンドウにメッセージを送信

タブコントロールを作成し、タブを選択します。




```
'=====================================
'= タブコントロールのタブを選択
'=    (CreateWindowEx4_3.bas)
'=====================================
#include "Windows.bi"

Type INITCOMMONCONTROLSEX
    dwSize      As Long
    dwICC       As Long
End Type
```

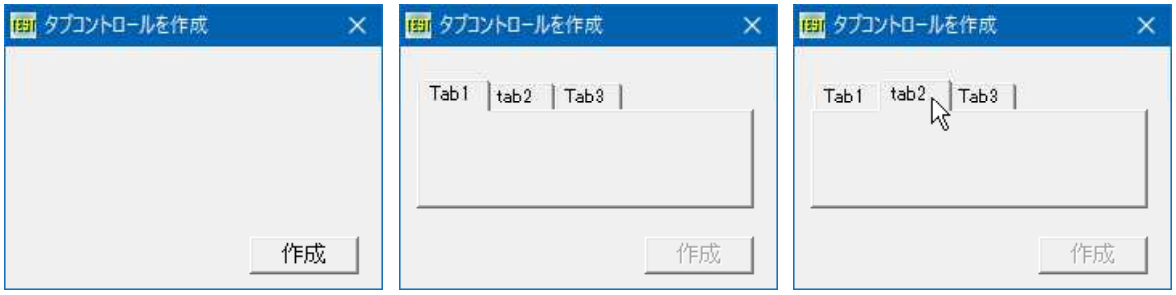
```
#define COLOR_BTNFACE 15                    '3Dオブジェクトの表面色
#define DEFAULT_GUI_FONT 17                 'ユーザーインターフェイス用のデフォルトフォント
#define ICC_TAB_CLASSES &H8                 'タブコントロール、ツールチップ
#define IDM_TAB1 &H100                      'TABコントロールのID
#define TCIF_TEXT &H1                       'pdzTextにデータが含まれる
#define TCM_INSERTITEMA &H1307              'タブ項目を追加
#define TCM_SETCURSEL &H130C                'タブを選択
#define WC_TABCONTROL "SysTabControl32"     'タブコントロール
#define WM_SETFONT &H30                     '論理フォントを設定する
#define WS_CHILD &H40000000                 '親ウィンドウを持つコントロール(子ウィンドウ)を作成する
#define WS_CLIPSIBLINGS &H4000000           '兄弟関係にある子ウィンドウをクリップする
#define WS_VISIBLE &H10000000               '可視状態のウィンドウを作成する
#define TCS_HOTTRACK &H40                   'マウスカーソルの下のタブを強調表示
#define TCS_TOOLTIPS &H4000                 'ツールヒントコントロールが作成されTTN_NEEDTEXTを発行

Type TCITEM
    mask            As Long
    dwState         As Long
    dwStateMask     As Long
    pszText         As Long
    cchTextMax      As Long
    iImage          As Long
    lParam          As Long
End Type

' システムの背景色を取得
Declare Function Api_GetSysColor& Lib "user32" Alias "GetSysColor" (ByVal nIndex&)

' ウィンドウ(コントロール)を作成
Declare Function Api_CreateWindowEx& Lib "user32" Alias "CreateWindowExA" (ByVal
ExStyle&, ByVal ClassName$, ByVal WinName$, ByVal Style&, ByVal x&, ByVal y&, ByVal
nWidth&, ByVal nHeight&, ByVal Parent&, ByVal Menu&, ByVal Instance&, Param&)

' ストックオブジェクトのハンドルを取得
Declare Function Api_GetStockObject& Lib "gdi32" Alias "GetStockObject" (ByVal nIndex&)

' CreateWindowExの解放
Declare Function Api_DestroyWindow& Lib "user32" Alias "DestroyWindow" (ByVal hWnd&)

' コモンコントロールライブラリからコモンコントロールのウィンドウクラスを登録して初期化
Declare Sub Api_InitCommonControls Lib "comctl32" Alias "InitCommonControls" ()

' ウィンドウにメッセージを送信
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

' ウィンドウにメッセージを送信
Declare Function Api_SendMessageByLong& Lib "user32" Alias "SendMessageA" (ByVal hWnd&,
ByVal wMsg&, ByVal wParam&, ByVal lParam&)

Var Shared Edit1 As Object
Var Shared Button1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var Shared hTabs As Long

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var rgbColor As Long

    'Buttonの表面色を取得(EDE9EC)
    rgbColor = Api_GetSysColor(COLOR_BTNFACE)

    'MainFormを取得色で塗り
    SetBackColor rgbColor
```

```
    '画面を消去
    Cls

    'MainFormを表示
    ShowWindow -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var icc As INITCOMMONCONTROLSEX
    Var tci As TCITEM
    Var index As Long
    Var Ret As Long

    'タブの選択
    index = Val(Edit1.GetWindowtext)
    If index > 3 Then Exit Sub

    icc.dwSize = Len(icc)
    icc.dwICC = ICC_TAB_CLASSES
    Api_InitCommonControls

    Ret = Api_DestroyWindow(hTabs)

    hTabs = Api_CreateWindowEx(0, WC_TABCONTROL, "", WS_CHILD Or WS_CLIPSIBLINGS Or
WS_VISIBLE Or TCS_HOTTRACK Or TCS_TOOLTIPS, 10, 20, 210, 80, GethWnd, IDM_TAB1, GethInst,
ByVal 0)

    If hTabs <> 0 Then
        Ret = Api_SendMessageByLong(hTabs, WM_SETFONT,
Api_GetStockObject(DEFAULT_GUI_FONT), 0)

        tci.mask = TCIF_TEXT
        tci.pszText = StrAdr("Tab1" & Chr$(0))
        Ret = Api_SendMessage(hTabs, TCM_INSERTITEMA, 0, tci)

        tci.mask = TCIF_TEXT
        tci.pszText = StrAdr("tab2" & Chr$(0))
        Ret = Api_SendMessage(hTabs, TCM_INSERTITEMA, 1, tci)

        tci.mask = TCIF_TEXT
        tci.pszText = StrAdr("Tab3" & Chr$(0))
        Ret = Api_SendMessage(hTabs, TCM_INSERTITEMA, 2, tci)

        tci.mask = TCIF_TEXT
        tci.pszText = StrAdr("Tab4" & Chr$(0))
        Ret = Api_SendMessage(hTabs, TCM_INSERTITEMA, 3, tci)
    End If

    Ret = Api_SendMessage(hTabs, TCM_SETCURSEL, index, ByVal 0)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
    Var Ret As Long

    Ret = Api_DestroyWindow(hTabs)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
```

```
Wend
Stop
End
```

タブコントロールをコードで作成（１）

GetSysColor システムの背景色を取得
CreateWindowEx ウィンドウ（コントロール）を作成
GetStockObject ストックオブジェクトのハンドルを取得
DestroyWindow CreateWindowExの解放
InitCommonControls コモンコントロールライブラリからコモンコントロールのウィンドウクラスを登録して初期化
SendMessage ウィンドウにメッセージを送信
SendMessageByLong ウィンドウにメッセージを送信




```
'=====================================
'= タブコントロールをコードで作成
'=    (CreateWindowEx4.bas)
'=====================================
#include "Windows.bi"

#define ICC_TAB_CLASSES &H8                 'タブコントロール、ツールチップ

Type INITCOMMONCONTROLSEX
    dwSize      As Long
    dwICC       As Long
End Type

#define TCIF_TEXT &H1                       'pdzTextにデータが含まれる
#define TCIF_IMAGE &H2                      'hImageにデータが含まれる
#define TCIF_STATE &H10                     'dwStateにデータが含まれる
#define TCIF_PARAM &H8                      'lParamにデータが含まれる

#define TCM_FIRST &H1300
#define TCM_INSERTITEMA (&H1300 + 7)
#define TCM_INSERTITEMW (&H1300 + 62)
#define TCM_INSERTITEM TCM_INSERTITEMA
#define TCM_GETCURSEL &H130B

#define WC_TABCONTROL "SysTabControl32"     'タブコントロール

#define WS_CHILD &H40000000                 '親ウィンドウを持つコントロール（子ウィンドウ）を作成する
#define WS_CLIPCHILDREN &H2000000           '親ウィンドウ内部を描画するとき、子ウィンドウが占める領域
                                            を除外
#define WS_CLIPSIBLINGS &H4000000           '兄弟関係にある子ウィンドウをクリップする
#define WS_OVERLAPPED &H0                   'オーバーラップウィンドウを作成する
#define WS_TABSTOP &H10000                  'ユーザーがTabキーを押すと入力フォーカスを受け取るコ
                                            ントロールを指定する
#define WS_VISIBLE &H10000000               '可視状態のウィンドウを作成する
#define WS_TABS (WS_CLIPCHILDREN Or WS_CLIPSIBLINGS Or WS_OVERLAPPED Or WS_VISIBLE Or
WS_CHILD)
#define WS_EX_LEFT &H0                      '左揃えされたプロパティを持つウィンドウを作成（デフォル
                                            ト）
#define WS_EX_LTRREADING &H0                '左から右への読み取り順序を持つプロパティを持ったウィ
                                            ンドウを作成（デフォルト）
#define WS_EX_RIGHTSCROLLBAR &H0            '垂直スクロールバーがクライアント領域の右側に置かれる
                                            （デフォルト）
```

```
#define WS_EX_TABS &H0                          '(WS_EX_LEFT Or WS_EX_LTRREADING Or
                                                WS_EX_RIGHTSCROLLBAR)
#define IDM_TAB1 &H100                          'TABコントロールのID

#define TCS_BOTTOM &H2                          'タブの位置を下にする
#define TCS_BUTTONS &H100                       'タブをプッシュボタンとして表示
#define TCS_FIXEDWIDTH &H400                    '全てのタブを同じサイズで表示
#define TCS_FLATBUTTONS &H8                     'タブ項目をフラットで表示
#define TCS_FOCUSNEVER &H8000                   'タブは入力フォーカスを受けない
#define TCS_FOCUSONBUTTONDOWN &H1000            'タブは、選択されと入力フォーカスを受ける
#define TCS_FORCEICONLEFT &H10                  'タブアイコンを左詰めにし、テキストをセンタリング
#define TCS_FORCELABELLEFT &H20                 'アイコンとテキストを左詰
#define TCS_HOTTRACK &H40                       'マウスカーソルの下のタブを強調表示
#define TCS_MULTILINE &H200                     '幅に対し、一行で全てのタブを表示できない場合タブを改
                                                行し、複数行で表示
#define TCS_OWNERDRAWFIXED &H2000               'オーナー描画タブ
#define TCS_RAGGEDRIGHT &H800                   'タブを左詰めで表示
#define TCS_RIGHT &H2                           'タブを右側に表示「TCS_VERTICAL」と同時に使用
#define TCS_RIGHTJUSTIFY 0                      '全てのタブの合計の幅が、タブコントロール全体のサイズ
                                                になるまで拡大
#define TCS_SCROLLOPPOSITE 1                    '選択されていないタブを反対に移動
#define TCS_SINGLELINE 0                        'タブが一行で表示されるデフォルトではこのスタイルが用
                                                いられる
#define TCS_TABS 0                              'タブはデフォルトスタイルで表示
#define TCS_TOOLTIPS &H4000                     'ツールヒントコントロールが作成されTTN_NEEDTEXTを発行
#define TCS_VERTICAL &H80                       'タブを右側に表示「TCS_RIGHT」と同時に使用

#define TCN_SELCHANGE -551                      '選択されているタブが変更されたことを通知
#define WM_SETFONT &H30                         '論理フォントを設定する
#define DEFAULT_GUI_FONT 17                     '
#define COLOR_BTNFACE 15                        '3Dオブジェクトの表面色

Type TCITEM
    mask            As Long
    dwState         As Long
    dwStateMask     As Long
    pszText         As Long
    cchTextMax      As Long
    iImage          As Long
    lParam          As Long
End Type

' システムの背景色を取得
Declare Function Api_GetSysColor& Lib "user32" Alias "GetSysColor" (ByVal nIndex&)

' ウィンドウ(コントロール)を作成
Declare Function Api_CreateWindowEx& Lib "user32" Alias "CreateWindowExA" (ByVal
ExStyle&, ByVal ClassName$, ByVal WinName$, ByVal Style&, ByVal x&, ByVal y&, ByVal
nWidth&, ByVal nHeight&, ByVal Parent&, ByVal Menu&, ByVal Instance&, Param&)

' ストックオブジェクトのハンドルを取得
Declare Function Api_GetStockObject& Lib "gdi32" Alias "GetStockObject" (ByVal nIndex&)

' CreateWindowExの解放
Declare Function Api_DestroyWindow& Lib "user32" Alias "DestroyWindow" (ByVal hWnd&)

' コモンコントロールライブラリからコモンコントロールのウィンドウクラスを登録して初期化
Declare Sub Api_InitCommonControls Lib "comctl32" Alias "InitCommonControls" ()

' ウィンドウにメッセージを送信
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

' ウィンドウにメッセージを送信
Declare Function Api_SendMessageByLong& Lib "user32" Alias "SendMessageA" (ByVal hWnd&,
ByVal wMsg&, ByVal wParam&, ByVal lParam&)

Var Shared Button1 As Object
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Var Shared hTabs As Long
```

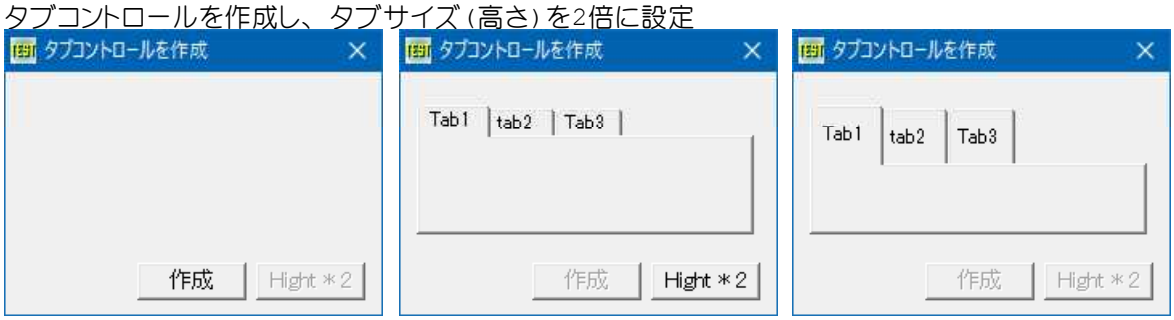
```
'=====================================
'=
'=====================================
Declare Sub Mainform_Start edecl ()
Sub Mainform_Start()
    Var rgbColor As Long

    'Buttonの表面色を取得(EDE9EC)
    rgbColor = Api_GetSysColor(COLOR_BTNFACE)

    'Mainformを取得色で塗り
    SetBackColor rgbColor

    '画面を消去し
    Cls

    'Mainformを表示
    ShowWindow -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var icc As INITCOMMONCONTROLSEX
    Var tci As TCITEM
    Var Ret As Long

    icc.dwSize = Len(icc)
    icc.dwICC = ICC_TAB_CLASSES
    Api_InitCommonControls

    hTabs = Api_CreateWindowEx(0, WC_TABCONTROL, "", WS_CHILD Or WS_CLIPSIBLINGS Or
WS_VISIBLE Or TCS_HOTTRACK Or TCS_TOOLTIPS, 10, 20, 210, 80, GethWnd, IDM_TAB1, GethInst,
ByVal 0)

    If hTabs <> 0 Then
        Ret = Api_SendMessageByLong(hTabs, WM_SETFONT,
Api_GetStockObject(DEFAULT_GUI_FONT), 0)

        tci.mask = TCIF_TEXT
        tci.pszText = StrAdr("Tab1" & Chr$(0))
        Ret = Api_SendMessage(hTabs, TCM_INSERTITEMA, 0, tci)

        tci.mask = TCIF_TEXT
        tci.pszText = StrAdr("tab2" & Chr$(0))
        Ret = Api_SendMessage(hTabs, TCM_INSERTITEMA, 1, tci)

        tci.mask = TCIF_TEXT
        tci.pszText = StrAdr("Tab3" & Chr$(0))
        Ret = Api_SendMessage(hTabs, TCM_INSERTITEMA, 2, tci)
    End If

    Button1.EnableWindow 0
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
    Var Ret As Long
    Ret = Api_DestroyWindow(hTabs)
End Sub

'=====================================
'=
'=====================================
While 1
```

```
    WaitEvent
Wend
Stop
End
```

タブコントロールをコードで作成（II）

GetSysColor システムの背景色を取得
CreateWindowEx ウィンドウ（コントロール）を作成
GetStockObject ストックオブジェクトのハンドルを取得
DestroyWindow CreateWindowExの解放
InitCommonControls コモンコントロールライブラリからコモンコントロールのウィンドウクラスを登録して初期化
SendMessage ウィンドウにメッセージを送信
SendMessageByLong ウィンドウにメッセージを送信

タブコントロールを作成し、タブサイズ（高さ）を2倍に設定




```
'=======================================
'= タブコントロールをコードで作成（II）
'=    (CreateWindowEx4_2.bas)
'=======================================
#include "Windows.bi"

#define ICC_TAB_CLASSES &H8                 'タブコントロール、ツールチップ

Type INITCOMMONCONTROLSEX
    dwSize      As Long
    dwICC       As Long
End Type

#define TCIF_TEXT &H1                       'pdzTextにデータが含まれる
#define TCM_FIRST &H1300
#define TCM_INSERTITEMA &H1307              '(&H1300 + 7)
#define TCM_SETITEMSIZE &H1329              '
#define TCM_GETITEMRECT &H130A              '

#define WC_TABCONTROL "SysTabControl32"     'タブコントロール
#define WS_CHILD &H40000000                 '親ウィンドウを持つコントロール（子ウィンドウ）を作成する
#define WS_CLIPSIBLINGS &H4000000           '兄弟関係にある子ウィンドウをクリップする
#define WS_VISIBLE &H10000000               '可視状態のウィンドウを作成する

#define TCS_HOTTRACK &H40                   'マウスカーソルの下のタブを強調表示
#define TCS_TOOLTIPS &H4000                 'ツールヒントコントロールが作成されTTN_NEEDTEXTを発行

#define WM_SETFONT &H30                     '論理フォントを設定する
#define DEFAULT_GUI_FONT 17                 '
#define COLOR_BTNFACE 15                    '3Dオブジェクトの表面色

Type TCITEM
    mask            As Long
    dwState         As Long
    dwStateMask     As Long
    pszText         As Long
    cchTextMax      As Long
    iImag           As Long
    lParam          As Long
End Type
```

```
Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

' システムの背景色を取得
Declare Function Api_GetSysColor& Lib "user32" Alias "GetSysColor" (ByVal nIndex&)

' ウィンドウ(コントロール)を作成
Declare Function Api_CreateWindowEx& Lib "user32" Alias "CreateWindowExA" (ByVal
ExStyle&, ByVal ClassName$, ByVal WinName$, ByVal Style&, ByVal x&, ByVal y&, ByVal
nWidth&, ByVal nHeight&, ByVal Parent&, ByVal Menu&, ByVal Instance&, Param&)

' ストックオブジェクトのハンドルを取得
Declare Function Api_GetStockObject& Lib "gdi32" Alias "GetStockObject" (ByVal nIndex&)

' CreateWindowExの解放
Declare Function Api_DestroyWindow& Lib "user32" Alias "DestroyWindow" (ByVal hWnd&)

' コモンコントロールライブラリからコモンコントロールのウィンドウクラスを登録して初期化
Declare Sub Api_InitCommonControls Lib "comctl32" Alias "InitCommonControls" ()

' ウィンドウにメッセージを送信
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

' ウィンドウにメッセージを送信
Declare Function Api_SendMessageByLong& Lib "user32" Alias "SendMessageA" (ByVal hWnd&,
ByVal wMsg&, ByVal wParam&, ByVal lParam&)

Var Shared Button1 As Object
Var Shared Button2 As Object

Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14

Var Shared hTabs As Long

'=======================================
'=
'=======================================
Declare Function MAKELPARAM(ByVal wHigh As Integer, ByVal wLow As Integer) As Long
Function MAKELPARAM(ByVal wHigh As Integer, ByVal wLow As Integer) As Long

    '下位BYTE値と16ビット左シフトした上位BYTE値の論理和
    MAKELPARAM = CLng(wLow) * (2 ^ 16) Or CLng(wHigh)
End Function

'=======================================
'=
'=======================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var rgbColor As Long

    'Buttonの表面色を取得(EDE9EC)
    rgbColor = Api_GetSysColor(COLOR_BTNFACE)

    'MainFormを取得色で塗り
    SetBackColor rgbColor

    '画面を消去し
    Cls

    'MainFormを表示
    ShowWindow -1

    Button2.EnableWindow 0
```

```
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var icc As INITCOMMONCONTROLSEX
    Var tci As TCITEM
    Var Ret As Long

    icc.dwSize = Len(icc)
    icc.dwICC = ICC_TAB_CLASSES

    Api_InitCommonControls

    hTabs = Api_CreateWindowEx(0, WC_TABCONTROL, "", WS_CHILD Or WS_CLIPSIBLINGS Or
WS_VISIBLE Or TCS_HOTTRACK Or TCS_TOOLTIPS, 10, 20, 210, 80, GethWnd, IDM_TAB1, GethInst,
ByVal 0)

    If hTabs <> 0 Then
        Ret = Api_SendMessageByLong(hTabs, WM_SETFONT,
Api_GetStockObject(DEFAULT_GUI_FONT), 0)

        tci.mask = TCIF_TEXT
        tci.pszText = StrAdr("Tab1" & Chr$(0))
        Ret = Api_SendMessage(hTabs, TCM_INSERTITEMA, 0, tci)

        tci.mask = TCIF_TEXT
        tci.pszText = StrAdr("tab2" & Chr$(0))
        Ret = Api_SendMessage(hTabs, TCM_INSERTITEMA, 1, tci)

        tci.mask = TCIF_TEXT
        tci.pszText = StrAdr("Tab3" & Chr$(0))
        Ret = Api_SendMessage(hTabs, TCM_INSERTITEMA, 2, tci)
    End If

    Button1.EnableWindow 0
    Button2.EnableWindow -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var lCurSel As Long
    Var rc As RECT
    Var X As Integer
    Var Y As Integer
    Var ItemSize As Long
    Var Ret As Long

    'タブ座標を取得するタブインデックスを指定
    lCurSel = 0

    '指定したインデックスのタブ座標を取得
    Ret = Api_SendMessage(hTabs, TCM_GETITEMRECT, lCurSel, rc)

    'タブ座標から幅を取得
    X = rc.Right - rc.Left

    '高さを取得し2倍
    Y = (rc.Bottom - rc.Top) * 2

    '取得したタブサイズからLPARAMを生成
    ItemSize = MAKELPARAM(X, Y)

    '新しいタブサイズを設定
    Ret = Api_SendMessage(hTabs, TCM_SETITEMSIZE, 0, ByVal ItemSize)
```

```
    Button2.EnableWindow 0
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
    Var Ret As Long

    Ret = Api_DestroyWindow(hTabs)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

タブコントロールをコードで作成（III）

GetSysColor システムの背景色を取得
CreateWindowEx ウィンドウ（コントロール）を作成
GetStockObject ストックオブジェクトのハンドルを取得
DestroyWindow CreateWindowExの解放
InitCommonControls コモンコントロールライブラリからコモンコントロールのウィンドウクラスを登録して初期化
SendMessage ウィンドウにメッセージを送信
SendMessageByLong ウィンドウにメッセージを送信

縦型タブコントロールを作成します。


```
'=====================================
'= タブコントロールをコードで作成（III）
'=    (CreateWindowEx4_6.bas)
'=====================================
#include "Windows.bi"

#define COLOR_BTNFACE 15                    '3Dオブジェクトの表面色
#define DEFAULT_GUI_FONT 17                 '
#define IDM_TAB1 &H100                      '
#define TCIF_TEXT &H1                       'pdzTextにデータが含まれる
#define TCM_INSERTITEMA &H1307              '(&H1300 + 7)
#define TCS_RIGHT &H2                       'タブを右側に表示「TCS_VERTICAL」と同時に使用
#define TCS_VERTICAL &H80                   'タブを右側に表示「TCS_RIGHT」と同時に使用
#define WM_SETFONT &H30                     '論理フォントを設定する
#define WC_TABCONTROL "SysTabControl32"     'タブコントロール
#define WS_CHILD &H40000000                 '親ウィンドウを持つコントロール（子ウィンドウ）を作成する
#define WS_CLIPSIBLINGS &H4000000           '兄弟関係にある子ウィンドウをクリップする
#define WS_VISIBLE &H10000000               '可視状態のウィンドウを作成する
```

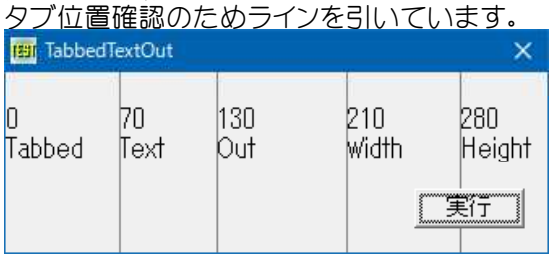
```
Type TCITEM
    mask            As Long
    dwState         As Long
    dwStateMask     As Long
    pszText         As Long
    cchTextMax      As Long
    iImage          As Long
    lParam          As Long
End Type

' システムの背景色を取得
Declare Function Api_GetSysColor& Lib "user32" Alias "GetSysColor" (ByVal nIndex&)

' ウィンドウ(コントロール)を作成
Declare Function Api_CreateWindowEx& Lib "user32" Alias "CreateWindowExA" (ByVal
ExStyle&, ByVal ClassName$, ByVal WinName$, ByVal Style&, ByVal x&, ByVal y&, ByVal
nWidth&, ByVal nHeight&, ByVal Parent&, ByVal Menu&, ByVal Instance&, Param&)

' ストックオブジェクトのハンドルを取得
Declare Function Api_GetStockObject& Lib "gdi32" Alias "GetStockObject" (ByVal nIndex&)

' CreateWindowExの解放
Declare Function Api_DestroyWindow& Lib "user32" Alias "DestroyWindow" (ByVal hWnd&)

' コモンコントロールライブラリからコモンコントロールのウィンドウクラスを登録して初期化
Declare Sub Api_InitCommonControls Lib "comctl32" Alias "InitCommonControls" ()

' ウィンドウにメッセージを送信
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

' ウィンドウにメッセージを送信
Declare Function Api_SendMessageByLong& Lib "user32" Alias "SendMessageA" (ByVal hWnd&,
ByVal wMsg&, ByVal wParam&, ByVal lParam&)

Var Shared Button1 As Object

Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var Shared hTabs As Long

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var rgbColor As Long

    'Buttonの表面色を取得
    rgbColor = Api_GetSysColor(COLOR_BTNFACE)

    'MainFormを取得色で塗り
    SetBackColor rgbColor

    '画面を消去し
    Cls

    'MainFormを表示
    ShowWindow -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var tci As TCITEM
    Var Ret As Long

    Api_InitCommonControls
```

```
    hTabs = Api_CreateWindowEx(0, WC_TABCONTROL, "", WS_CHILD Or WS_CLIPSIBLINGS Or
WS_VISIBLE Or TCS_VERTICAL Or TCS_RIGHT, 10, 20, 210, 150, GethWnd, IDM_TAB1, GethInst,
ByVal 0)

    If hTabs <> 0 Then
        Ret = Api_SendMessageByLong(hTabs, WM_SETFONT,
Api_GetStockObject(DEFAULT_GUI_FONT), 0)

        tci.mask = TCIF_TEXT
        tci.pszText = StrAdr("Tab1" & Chr$(0))
        Ret = Api_SendMessage(hTabs, TCM_INSERTITEMA, 0, tci)

        tci.mask = TCIF_TEXT
        tci.pszText = StrAdr("tab2" & Chr$(0))
        Ret = Api_SendMessage(hTabs, TCM_INSERTITEMA, 1, tci)

        tci.mask = TCIF_TEXT
        tci.pszText = StrAdr("Tab3" & Chr$(0))
        Ret = Api_SendMessage(hTabs, TCM_INSERTITEMA, 2, tci)
    End If

    Button1.EnableWindow 0
End Sub

'=======================================
'=
'=======================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
    Var Ret As Long

    Ret = Api_DestroyWindow(hTabs)
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

タブ文字を含むテキストを出力

タブ文字を含むテキストを出力します。
GetSysColor システムの背景色を取得
SetBkColor デバイスコンテキストの背景色を設定
TabbedTextOut タブ文字を含むテキストを出力する
GetDC デバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放

例では、システムの3Dオブジェクト表面色(Buttonの色)を取得し、フォームの色としています。
タブ位置確認のためラインを引いています。




```
'=======================================
'= タブ文字を含むテキストを出力
'=    (TabbedTextOut.bas)
'=======================================
```

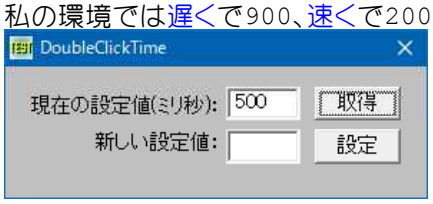
```
#include "Windows.bi"

' システムの背景色を取得
Declare Function Api_GetSysColor& Lib "user32" Alias "GetSysColor" (ByVal nIndex&)

' デバイスコンテキストの背景色を設定
Declare Function Api_SetBkColor& Lib "gdi32" Alias "SetBkColor" (ByVal hDC&, ByVal
crColor&)

' タブ文字を含むテキストを出力する
Declare Function Api_TabbedTextOut& Lib "user32" Alias "TabbedTextOutA" (ByVal hDC&,
ByVal x&, ByVal y&, ByVal lpString$, ByVal nCount&, ByVal nTabPositions&,
lpnTabStopPositions&, ByVal nTabOrigin&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

#define COLOR_BTNFACE 15                    '3Dオブジェクトの表面色

Var Shared Button1 As Object

Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var Shared hDC As Long

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var BkColor As Long
    Var Ret As Long

    'デバイスコンテキスト取得
    hDC = Api_GetDC(GethWnd)

    'システムの3Dオブジェクトの表面色取得
    BkColor = Api_GetSysColor(COLOR_BTNFACE)

    '背景色を設定
    Ret = Api_SetBkColor(hDC, BkColor)
    SetBackColor BkColor
    Cls
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var TabStop(3) As Long
    Var mStr As String
    Var Ret As Long

    'タブストップを設定
    TabStop(0) = 70
    TabStop(1) = 130
    TabStop(2) = 210
    TabStop(3) = 280

    '1行目
    mStr = "0" & Chr$(9) & Trim$(Str$(TabStop(0))) & Chr$(9) & Trim$(Str$(TabStop(1))) &
Chr$(9) & Trim$(Str$(TabStop(2))) & Chr$(9) & Trim$(Str$(TabStop(3)))
    Ret = Api_TabbedTextOut(hDC, 0, 20, mStr, Len(mStr), 4, TabStop(0), 0)

    '2行目
    mStr = "Tabbed" & Chr$(9) & "Text" & Chr$(9) & "Out" & Chr$(9) & "Width" & Chr$(9) &
```

```
"Height"
    Ret = Api_TabbedTextOut(hDC, 0, GetTextHeight(mstr) + 20, mStr, Len(mStr), 4, TabStop
(0), 0)

    For x = 0 To 3
        Line(TabStop(x), 0)-(TabStop(x), 140),, 1
    Next
End Sub

'=======================================
'=
'=======================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
    Ret = Api_ReleaseDC(GethWnd, hDC)
    End
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

ダブルクリック時間の取得と設定

マウスのダブルクリック時間の取得と設定を実行します。
GetDoubleClickTime ダブルクリック時間の取得
SetDoubleClickTime ダブルクリック時間の設定

私の環境では遅くで900、速くで200でした。



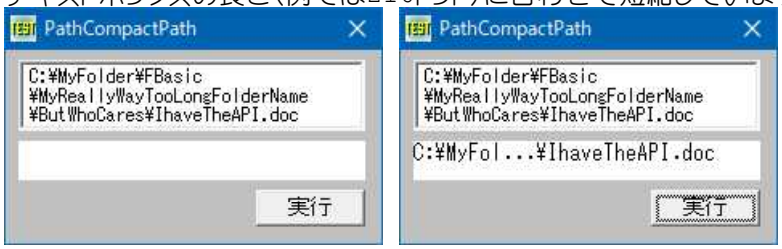

```
'=======================================
'=  ダブルクリック時間の取得と設定
'=    (DoubleClickTime.bas)
'=======================================
#include "Windows.bi"

' ダブルクリック時間の取得
Declare Function Api_SetDoubleClickTime& Lib "user32" Alias "SetDoubleClickTime" (ByVal
wCount&)

' ダブルクリック時間の取得
Declare Function Api_GetDoubleClickTime& Lib "user32" Alias "GetDoubleClickTime" ()

Var Shared Text2 As Object
```

```
Text2.Attach GetDLgItem("Text2")
Text2.SetFontSize 14

Var Shared Dct As Long                          'ダブルクリック時間

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Dct = Api_GetDoubleClickTime                'ダブルクリック時間取得
    Text2.SetWindowText Str$(Dct)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Ret As Long

    Dct = val(GetDlgItemText("Edit1"))

    If Dct = 0 Then Exit Sub
    Ret = Api_SetDoubleClickTime(Dct)           'ダブルクリック時間設定
    Text2.SetWindowText ""
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

短縮形式のパスを取得

PathCompactPath 短縮形式のパスを取得
GetDC デバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放

テキストボックスの長さ(例では216ドット)に合わせて短縮しています。

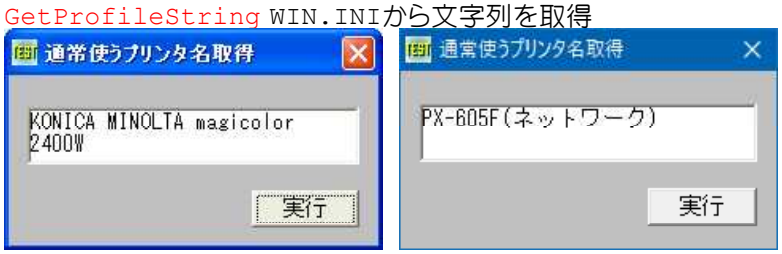



```
'=====================================
'= 短縮形式のパスを取得
'=    (PathCompactPath.bas)
'=====================================
#include "Windows.bi"

' 短縮形式のパスを取得
Declare Function Api_PathCompactPath& Lib "shlwapi" Alias "PathCompactPathA" (ByVal
hDC&, ByVal lpszPath$, ByVal dx&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)
```

```
' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

Var Shared Edit1 As Object
Var Shared Text1 As Object
Var Shared Button1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 12
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=======================================
'=
'=======================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Edit1.SetWindowText
"C:¥MyFolder¥FBasic¥MyReallyWayTooLongFolderName¥ButWhoCares¥IhaveTheAPI.doc"
End Sub

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hDC As Long
    Var CtlWidth As Long
    Var FileName As String

    FileName = Edit1.GetWindowtext

    Text1.SetWindowSize 216, 24
    CtlWidth = Text1.GetWidth
    hDC = Api_GetDC(Text1.GethWnd)

    Ret = Api_PathCompactPath(hDC, FileName, CtlWidth )
    Text1.SetWindowText FileName

    Ret = Api_ReleaseDC(hDC, Text1.GethWnd)
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

通常使うプリンタ名を取得

iniファイルから「通常使うプリンタ」名を取得します。
GetProfileString WIN.INIから文字列を取得



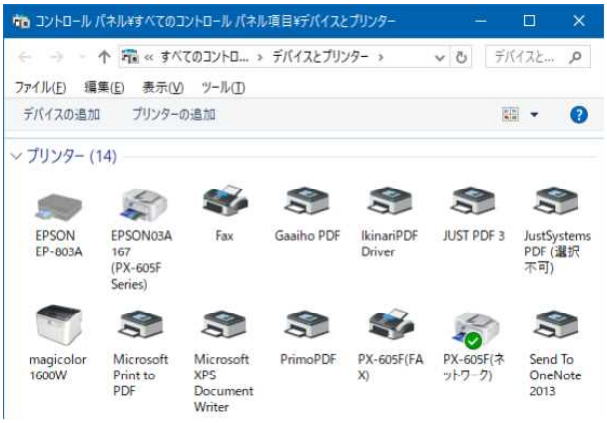

```
'=======================================
'= 通常使うプリンタ名を取得
'=   (GetDefaultPrinter.bas)
'=======================================
```

```
#include "Windows.bi"

' WIN.INIから文字列を取得
Declare Function Api_GetProfileString& Lib "Kernel32" Alias "GetProfileStringA" (ByVal
lpAppName$, ByVal lpKeyName$, ByVal lpDefault$, ByVal lpReturnedString$, ByVal nSize&)

Var Shared Text1 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var SectionName As String
    Var KeyName As String
    Var DefaultStr As String
    Var Buffer As String * 1024
    Var Length As Long
    Var RtnLength As Long

    'セクションを指定
    SectionName = "windows"

    'キーを指定
    KeyName = "device"

    'デフォルト値を指定
    DefaultStr = ",,,"

    'バッファの長さを指定
    Length = Len(Buffer)

    '値を取得
    RtnLength = Api_GetProfileString(SectionName, KeyName, DefaultStr, Buffer, Length)

    '取得した値からプリンタ名を表示
    Text1.SetWindowtext Left$(Buffer, InStr(Buffer, ",") - 1)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

通常使うプリンタを設定

GetProfileString WIN.INIから文字列を取得
WriteProfileString WIN.INIの指定のセクションの内容を変更
SendNotifyMessage 指定されたメッセージを1つまたは複数のウィンドウへ送信

リストボックスで選択し、設定をクリックすると「通常使うプリンタ」に設定されます。



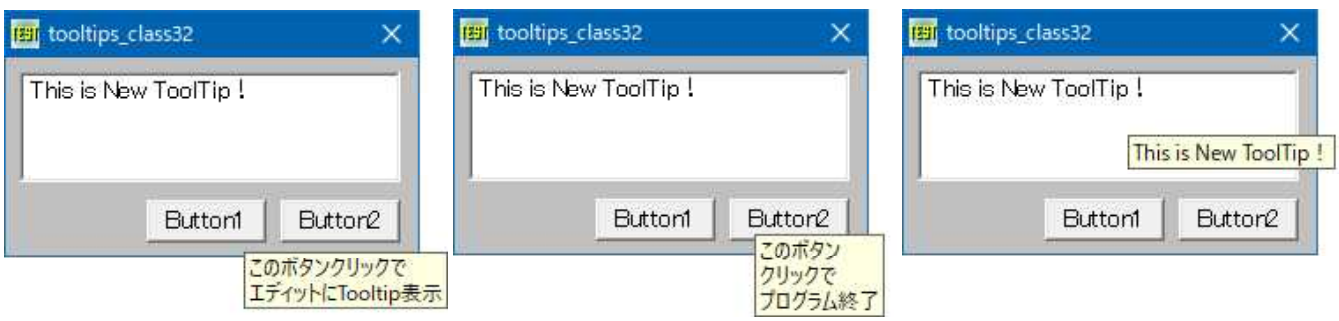

```
'=======================================
'= 通常使うプリンタを設定
'=    (WriteProfileString2.bas)
'=======================================
#include "Windows.bi"

#define HWND_BROADCAST &HFFFF                'トップレベルウィンドウに対してメッセージを送る
#define WM_WININICHANGE &H1A                 'WIN.INIが変更された

' WIN.INIから文字列を取得
Declare Function Api_GetProfileString& Lib "Kernel32" Alias "GetProfileStringA" (ByVal
lpAppName$, ByVal lpKeyName$, ByVal lpDefault$, ByVal lpReturnedString$, ByVal nSize&)

' WIN.INIの指定のセクションの内容を変更
Declare Function Api_WriteProfileString& Lib "Kernel32" Alias "WriteProfileStringA"
(ByVal lpszSection$, ByVal lpszKeyName$, ByVal lpszString$)

' 指定されたメッセージを1つまたは複数のウィンドウへ送信
Declare Function Api_SendNotifyMessage& Lib "user32" Alias "SendNotifyMessageA" (ByVal
hWnd&, ByVal msg&, ByVal wParam&, lParam As Any)

Var Shared List1 As Object
Var Shared Button1 As Object

List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=======================================
'=
'=======================================
Declare Sub SetDefPrinter(PrinterName As String, DriverName As String, PrinterPort As
String)
Sub SetDefPrinter(PrinterName As String, DriverName As String, PrinterPort As String)
    Var DeviceLine As String
    Var Ret As Long

    DeviceLine = PrinterName & "," & DriverName & "," & PrinterPort

    Ret = Api_WriteProfileString("windows", "Device", DeviceLine)
    Ret = Api_SendNotifyMessage(HWND_BROADCAST, WM_WININICHANGE, 0, "windows")
End Sub

'=======================================
'=
'=======================================
Declare Sub GetDriverAndPort(Buffer As String, DriverName As String, PrinterPort As
String)
Sub GetDriverAndPort(Buffer As String, DriverName As String, PrinterPort As String)
    Var posDriver As Long
    Var posPort As Long
    DriverName = ""
    PrinterPort = ""
```

```
    posDriver = InStr(Buffer, ",")

    If posDriver > 0 Then
        DriverName = Left$(Buffer, posDriver - 1)
        posPort = InStr(posDriver + 1, Buffer, ",")

        If posPort > 0 Then
            PrinterPort = Mid$(Buffer, posDriver + 1, posPort - posDriver - 1)
        End If
    End If
End Sub

'====================================
'=
'====================================
Declare Function StripNulls(startstr As String) As String
Function StripNulls(startstr As String) As String
    Var epos As Long

    epos = InStr(startstr, Chr$(0))

    If epos Then
        StripNulls = Mid$(startstr, 1, epos - 1)
        startstr = Mid$(startstr, epos + 1, Len(startstr))
    End If
End Function

'====================================
'=
'====================================
Declare Function ProfileLoadWinIniList(lpSectionName As String) As Long
Function ProfileLoadWinIniList(lpSectionName As String) As Long
    Var success As Long
    Var nSize As Long
    Var lpKeyName As String
    Var Str As String
    Var Ret As Long

    Str = Space$(8102)
    nSize = Len(Str)
    success = Api_GetProfileString(lpSectionName, ByVal 0, ByVal 0, Str, nSize)

    If success Then
        Str = Left$(Str, success)

        Do Until Str = ""
            lpKeyName = StripNulls(Str)
            List1.AddString lpKeyName
        Loop
    End If

    ProfileLoadWinIniList = List1.GetCount
End Function

'====================================
'=
'====================================
Declare Sub SetDefaultPrinterWinNT()
Sub SetDefaultPrinterWinNT()
    Var Buffer As String
    Var DeviceName As String
    Var DriverName As String
    Var PrinterPort As String
    Var PrinterName As String
    Var r As Long

    If List1.GetCount > -1 Then
        Buffer = Space$(1024)
        PrinterName = List1.GetText(List1.GetCursel)
```

```
        Ret = Api_GetProfileString("PrinterPorts", PrinterName, "", Buffer, Len(Buffer))

        GetDriverAndPort Buffer, DriverName, PrinterPort

        If (Len(DriverName) > 0) And (Len(PrinterPort) > 0) Then
            SetDefPrinter PrinterName, DriverName, PrinterPort
        End If
    End If
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var Ret As Long

    Ret = ProfileLoadWinIniList("PrinterPorts")
    Button1.EnableWindow 0
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
   SetDefaultPrinterWinNT
End Sub

'=====================================
'=
'=====================================
Declare Sub List1_Click edecl ()
Sub List1_Click()
    Button1.EnableWindow -1
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

ツールチップの作成（１）

CreateWindowEx ウィンドウ（コントロール）を作成
DestroyWindow CreateWindowExの解放
InitCommonControls コモンコントロールのダイナミックリンクライブラリ（DLL）に含まれている、特定のコモンコントロールクラスを登録
SendMessage ウィンドウにメッセージを送信
SetWindowPos ウィンドウのサイズ、位置、および Z オーダーを設定

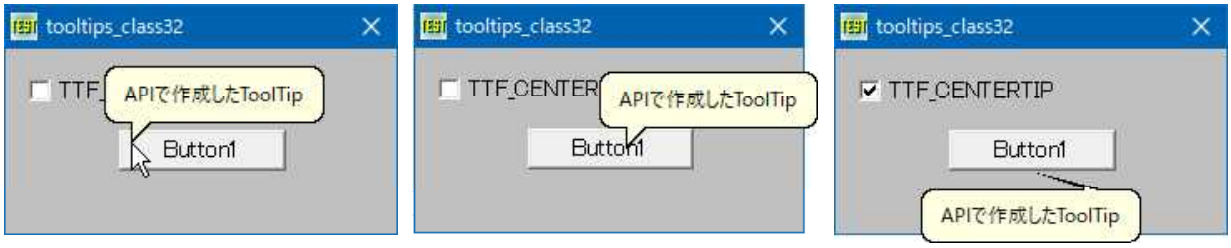


```
'====================================
'= ツールチップの作成(Ⅰ)
'=    (ToolTip.bas)
'====================================
#include "Windows.bi"

Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

Type TOOLINFO
    cbSize      As Long
    uFlags      As Long
    hwnd        As Long
    uId         As Long
    cRect       As RECT
    hinst       As Long
    lpszText    As Long
End Type

' ウィンドウ(コントロール)を作成
Declare Function Api_CreateWindowEx& Lib "user32" Alias "CreateWindowExA" (ByVal
ExStyle&, ByVal ClassName$, ByVal WinName$, ByVal Style&, ByVal x&, ByVal y&, ByVal
nWidth&, ByVal nHeight&, ByVal Parent&, ByVal Menu&, ByVal Instance&, Param As Any)

' CreateWindowExの解放
Declare Function Api_DestroyWindow& Lib "user32" Alias "DestroyWindow" (ByVal hWnd&)

' コモンコントロールのダイナミックリンクライブラリ(DLL)に含まれている、特定のコモンコントロールクラスを登録
Declare Sub Api_InitCommonControls Lib "comctl32" Alias "InitCommonControls" ()

' ウィンドウにメッセージを送信
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

' ウィンドウのサイズ、位置、および Z オーダーを設定
Declare Function Api_SetWindowPos& Lib "user32" Alias "SetWindowPos" (ByVal hWnd&, ByVal
hWndInsertAfter&, ByVal X&, ByVal Y&, ByVal CX&, ByVal CY&, ByVal uFlags&)

#define WM_USER &H400                       'ユーザーが定義できるメッセージの使用領域を表すだけで
                                            これ自体に意味はない
#define TTM_ADDTOOL (&H400 + 4)             'ツールチップをツールに登録
#define TTM_SETMAXTIPWIDTH (&H400 + 24)     'ツールチップの最大幅
#define TTF_CENTERTIP &H2                   'チップをツールの中心に表示
#define TTF_IDISHWND &H1                    'uIdメンバは、ツールのウィンドウハンドル
#define TTF_RTLREADING &H4                  'テキストを右から左に表示(ヘブライ語、またはアラビア語)
#define TTF_SUBCLASS &H10                   'ツールをサブクラス化してメッセージを取得
#define HWND_TOPMOST (-1)                   'ウィンドウを常に最前面に配置
#define SWP_NOMOVE &H2                      'ウィンドウの現在位置を保持する
#define SWP_NOSIZE &H1                      'ウィンドウの現在のサイズを保持する
#define SWP_NOACTIVATE &H10                 'ウィンドウをアクティブにしない

Var Shared MainForm As Object
Var Shared Edit1 As Object
Var Shared Button1 As Object
Var Shared Button2 As Object

MainForm.Attach GethWnd
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14

Var Shared hToolTip As Long

'====================================
'=
'====================================
```

```
Declare Sub SetMultiLineToolTip(ByVal hwnd As Long, sToolTip As String)
Sub SetMultiLineToolTip(ByVal hwnd As Long, sToolTip As String)
    Var ti As TOOLINFO

    ti.cbSize = Len(ti)
    ti.hwnd = hWnd
    ti.uFlags = TTF_IDISHWND Or TTF_SUBCLASS
    ti.uId = hWnd
    ti.hinst = GethInst
    ti.lpszText = StrAdr(sToolTip & Chr$(0))

    Ret = Api_SendMessage(hToolTip, TTM_ADDTOOL, 0, ti)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var Ret As Long

    Api_InitCommonControls

    hToolTip = Api_CreateWindowEx(0, "tooltips_class32", "", 0, 0, 0, 0, 0, 0, 0, 0, 0)

    Ret = Api_SendMessage(hToolTip, TTM_SETMAXTIPWIDTH, 0, 300)

    SetMultiLineToolTip Button1.GethWnd, "このボタンクリックで" & Chr$(13, 10) & "エディットに
Tooltip表示"
    SetMultiLineToolTip Button2.GethWnd, "このボタン" & Chr$(13, 10) & "クリックで" & Chr$(13,
10) & "プログラム終了"
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    SetMultiLineToolTip Edit1.GethWnd, Edit1.GetWindowText
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Ret As Long

    Ret = Api_DestroyWindow(hToolTip)
    End
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_QueryClose edecl ()
Sub mainForm_QueryClose()
    Var Ret As Long
    Ret = Api_DestroyWindow(hToolTip)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

ツールチップの作成(II)

CreateWindowEx ウィンドウ(コントロール)を作成
DestroyWindow CreateWindowExの解放
InitCommonControls コモンコントロールのダイナミックリンクライブラリ(DLL)に含まれている、特定のコモンコントロールクラスを登録
SendMessage ウィンドウにメッセージを送信
SetWindowPos ウィンドウのサイズ、位置、および Z オーダーを設定




```
'=======================================
'= ツールチップの作成(II)
'=    (ToolTip2.bas)
'=======================================
#include "Windows.bi"

Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

Type TOOLINFO
    cbSize      As Long
    uFlags      As Long
    hwnd        As Long
    uId         As Long
    rc          As RECT
    hInst       As Long
    lpszText    As Long
    lParam      As Long
End Type

' ウィンドウ(コントロール)を作成
Declare Function Api_CreateWindowEx& Lib "user32" Alias "CreateWindowExA" (ByVal
ExStyle&, ByVal ClassName$, ByVal WinName$, ByVal Style&, ByVal x&, ByVal y&, ByVal
nWidth&, ByVal nHeight&, ByVal Parent&, ByVal Menu&, ByVal Instance&, ByVal Param&)

' ウィンドウのサイズ、位置、および Z オーダーを設定
Declare Function Api_SetWindowPos& Lib "user32" Alias "SetWindowPos" (ByVal hWnd&, ByVal
hWndInsertAfter&, ByVal X&, ByVal Y&, ByVal CX&, ByVal CY&, ByVal uFlags&)

' ウィンドウにメッセージを送信。この関数は、指定したウィンドウのウィンドウプロシージャが処理を終了するまで制御を返さない
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)
' ウィンドウのクライアント領域の座標を取得
Declare Function Api_GetClientRect& Lib "user32" Alias "GetClientRect" (ByVal hWnd&,
lpRect As RECT)

' CreateWindowExの解放
Declare Function Api_DestroyWindow& Lib "user32" Alias "DestroyWindow" (ByVal hWnd&)

#define TOOLTIPS_CLASSA "tooltips_class32"
#define TTF_IDISHWND &H1                    'uIdメンバは、ツールのウィンドウハンドル
#define TTF_CENTERTIP &H2                   'チップをツールの中心に表示
#define TTF_SUBCLASS &H10                   'ツールをサブクラス化してメッセージを取得
#define TTM_ADDTOOL &H404                   '(WM_USER+4)ツールの追加登録
#define TTS_BALLOON &H40                    'バルーンチップを作る
```

```
Var Shared hWnd As Long
Var Shared flag As Integer

Var Shared Button1 As Object
Var Shared Check1 As Object

Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Check1.Attach GetDlgItem("Check1") : Check1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub ToolTip ()
Sub ToolTip()
    Var ti As TOOLINFO
    Var Ret As Long

    ti.cbSize = Len(ti)
    flag = Check1.GetCheck
    If flag = 0 Then
        ti.uFlags = TTF_IDISHWND Or TTF_SUBCLASS
    Else
        ti.uFlags = TTF_IDISHWND Or TTF_SUBCLASS Or TTF_CENTERTIP
    End If
    ti.hwnd = GethWnd
    ti.hinst = GethInst
    ti.uid = Button1.GethWnd
    ti.lpszText = StrAdr("APIで作成したToolTip" & Chr$(0))

    hWnd = Api_CreateWindowEx(0, TOOLTIPS_CLASSA, "", TTS_BALLOON, 0, 0, 0, 0, 0, 0, 0, 0)

    Ret = Api_SendMessage(hWnd, TTM_ADDTOOL, 0, ti)
End Sub

'=====================================
'=
'=====================================
Declare Sub DestroyTP ()
Sub DestroyTP()
    Var Ret As Long

    Ret = Api_DestroyWindow(hWnd)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    ToolTip
End Sub

'=====================================
'=
'=====================================
Declare Sub Check1_on edecl ()
Sub Check1_on()
    DestroyTP
    ToolTip
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
    DestroyTP
End Sub
```

```
'====================================
'=
'====================================
While 1
    WaitEvent
Wend
Stop
End
```

参考

```
#define TTDT_AUTOMATIC 0            '遅延時間の割合をデフォルトに戻す
#define TTDT_AUTOPOP 2              'ツールチップコントロールの表示時間
#define TTDT_INITIAL 3              'カーソルを置いてから表示されるまでの時間を指定
#define TTDT_RESHOW 1               '

#define TTF_ABSOLUTE &H80           '
#define TTF_CENTERTIP &H2           'チップをツールの中心に表示
#define TTF_DI_SETITEM &H8000       '
#define TTF_IDISHWND &H1            'uIdメンバは、ツールのウィンドウハンドル
#define TTF_RTLREADING &H4          'テキストを右から左に表示(ヘブライ語、またはアラビア語)
#define TTF_SUBCLASS &H10           'ツールをサブクラス化してメッセージを取得
#define TTF_TRACK &H20              '
#define TTF_TRANSPARENT &H100       '

#define TTM_ACTIVATE &H401          '(WM_USER+1)アクティブ・インアクティブの切り替え
#define TTM_ADDTOOL &H404           '(WM_USER+4)ツールの追加登録
#define TTM_ADJUSTRECT &H431        '(WM_USER+31)
#define TTM_DELTOOLA &H405          '(WM_USER+5)ツールの登録削除
#define TTM_DELTOOLW &H458          '(WM_USER+51)ツールの登録削除
#define TTM_ENUMTOOLSA &H401        '(WM_USER+14)登録ツール情報の取得(列挙)
#define TTM_ENUMTOOLSW &H458        '(WM_USER+58)登録ツール情報の取得(列挙)
#define TTM_GETBUBBLESIZE &H430     '(WM_USER+30)
#define TTM_GETCURRENTTOOLA &H415   '(WM_USER+15)「現在のツール」の情報取得
#define TTM_GETCURRENTTOOLW &H459   '(WM_USER+59)「現在のツール」の情報取得
#define TTM_GETDELAYTIME &H421      '(WM_USER+21)遅延時間の取得
#define TTM_GETMARGIN &H427         '(WM_USER+27)マージンの取得
#define TTM_GETMAXTIPWIDTH &H425    '(WM_USER+25)チップウィンドウの最大幅の取得
#define TTM_GETTEXT &H411           '(WM_USER+11)登録テキストの取得
#define TTM_GETTIPTEXTCOLOR &H423   '(WM_USER+23)テキスト色の取得
#define TTM_GETTOOLCOUNT &H413      '(WM_USER+13)登録ツールの数を取得
#define TTM_GETTOOLINFO &H408       '(WM_USER+8)登録ツール情報の取得
#define TTM_NEWTOOLRECT &H406       '(WM_USER+6)ツール範囲の再設定
#define TTM_POP &H428               '(WM_USER+28)強制消去
#define TTM_RELAYEVENT &H407        '(WM_USER+7)マウスメッセージを中継
#define TTM_SETDELAYTIME &H403      '(WM_USER+3)遅延時間の設定
#define TTM_SETMARGIN &H426         '(WM_USER+26)マージンの設定
#define TTM_SETMAXTIPWIDTH &H424    '(WM_USER+24)チップウィンドウの最大幅の設定
#define TTM_SETTIPBKCOLOR &H419     '(WM_USER+19)背景色の設定
#define TTM_SETTIPTEXTCOLOR &H420   '(WM_USER+20)テキスト色の設定
#define TTM_SETTOOLINFO &H409       '(WM_USER+9)登録ツールの情報を再設定
#define TTM_UPDATE &H429            '(WM_USER+29)強制再描画
#define TTM_UPDATETIPTEXT &H412     '(WM_USER+12)登録テキストの再設定
#define TTM_WINDOWFROMPOINT &H416   '(WM_USER+16)

#define TTS_ALWAYSTIP 1             'ツールチップコントロールの親ウィンドウが非アクティブでも常に
                                    表示される
#define TTS_BALLOON &H40            'バルーンチップを作る
#define TTS_CLOSE &H80              'タイトルをツールチップに付けたときに右上に×ボタンが表示
#define TTS_NOANIMATE &H10          'アニメーション表示をしない
#define TTS_NOFADE &H20             '消えるときフェードアウトしない
#define TTS_NOPREFIX 2              '文字列にアンパサント(&)が入っていると、普通これを無視しま
                                    すが、このスタイルを指定すると表示
#define TTS_USEVISUALSTYLE &H100    'ハイパーリンク付きの文字列を挿入するときに指定
```

月数･日数計算

開始年月日～終了年月日をyyyymmdd(19980512)のように入力し、日数計算ボタンをクリックします。
月(1～12)、日(1～31)の範囲をはずれた場合再入力、月末日はEndOfMonth部でチェックしており、
月末日を超えた数字を入力した場合は修正されます。例:20040230 → 20040229
丸1ヶ月、丸1日を表しており開始月、開始日は計算に入っていません。


```
'=====================================
'= 月数･日数計算
'=    (DaysCalc.bas)
'=====================================
#include "Windows.bi"

Var shared Text(5) As Object
Var shared Edit(1) As Object
Var shared Button1 As Object

For i = 0 To 5
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1)))
    Text(i).SetFontSize 14
Next
For i = 0 To 1
    Edit(i).Attach GetDlgItem("Edit" & Trim$(Str$(i + 1)))
    Edit(i).SetFontSize 14
Next
Button1.Attach GetDlgItem("Button1")
Button1.SetFontSize 14

' 月数･日数判定用
def fnN(Y, M, D) = Int(365.25 * Y) + Int(Y / 400) - int(Y / 1000) + Int(30.59 * (M - 2)) + D +
678912!
def fnL(Y1, M1, D1, Y2, M2, D2) = fnN(Y2, M2, D2) - fnN(Y1, M1, D1)

Var shared SY1 As Long                          '指定開始年
Var shared SM1 As Long                          '    〃   月
Var shared SD1 As Long                          '    〃   日
Var shared SY2 As Long                          '指定終了年
Var shared SM2 As Long                          '    〃   月
Var shared SD2 As Long                          '    〃   日
Var shared SY As Long                           '月末日用年
Var shared SM As Long                           '    〃   月
Var shared SD As Long                           '    〃   日
Var shared SMM As Long                          '月数(結果)
Var shared SDD As Long                          '日数(結果)
Var shared SE As Long                           '月末日(結果)
Var shared WYM As Long                          '月数ワーク
Var shared CrLf As String


'=====================================
'= 月末(月の最終日取得)
'=====================================
Declare Sub EndOfMonth edecl ()
Sub EndOfMonth()
    If SM <> 2 Then SE = Abs(7.5 - SM) Mod 2 + 30 : Exit Sub

    If SY Mod 4 <> 0 Then
        SE = 28
```

```
    Else If SY Mod 100 <> 0 Then
        SE = 29
    Else If SY Mod 400 <> 0 Then
        SE = 28
    Else
        SE = 29
    End If
End Sub

'=====================================
'= 日数計算
'=====================================
Declare Sub DaysCalc edecl ()
Sub DaysCalc()
    If SD1 = 0 Or SD2 = 0 Then Exit Sub

    ZYM = WYM
    WYM = (SY2 -SY1 ) * 12
    SMM = WYM + SM2 - SM1
    If SD1 > SD2 Then
        SY = SY2
        SM = SM2
        EndOfMonth
        If SE <> SD2 Then
            SMM = SMM - 1
        End If
    End If

    SM3 = Int(SMM / 6)
    SY5 = SY1 + Int(SM3 / 2)
    SM5 = SM1 + (SM3 Mod 2) * 6
    If SM5 > 12 Then
        SY5 = SY5 + 1
        SM5 = SM5 - 12
    End If

    SY = SY5
    SM = SM5
    EndOfMonth
    SD5 = SE
    If SD5 > SD1 Then SD5 = SD1
    WY1 = SY1
    WM1 = SM1
    WY2 = SY2
    WM2 = SM2
    WY5 = SY5
    WM5 = SM5
    If SM1 = 1 Or SM1 = 2 Then WM1 = SM1 + 12 : WY1 = SY1 - 1
    If SM2 = 1 Or SM2 = 2 Then WM2 = SM2 + 12 : WY2 = SY2 - 1
    If SM5 = 1 Or SM5 = 2 Then WM5 = SM5 + 12 : WY5 = SY5 - 1
    SDN = fnL(WY5, WM5, SD5, WY2, WM2, SD2)
    SDD = fnL(WY1, WM1, SD1, WY2, WM2, SD2)
    WYM = ZYM
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
sub MainForm_Start()
    CrLf = Chr$(13, 10)
    Edit(1).SetFocus
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_On edecl ()
sub Button1_On()
    SY1 = Val(Left$(GetDlgItemText("Edit1"), 4))
```

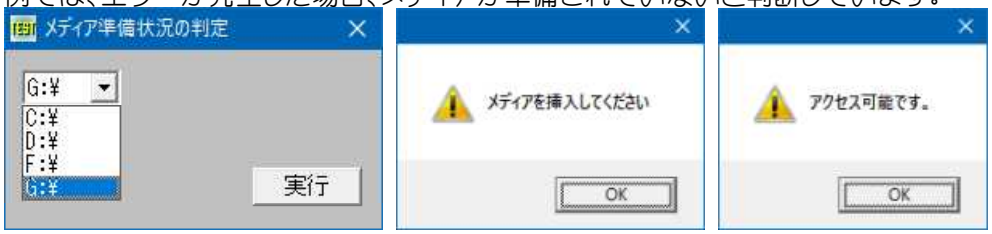
```
    SM1 = Val(Mid$(GetDlgItemText("Edit1"), 5, 2))
    SD1 = Val(Right$(GetDlgItemText("Edit1"), 2))

    SY2 = Val(Left$(GetDlgItemText("Edit2"), 4))
    SM2 = Val(Mid$(GetDlgItemText("Edit2"), 5, 2))
    SD2 = Val(Right$(GetDlgItemText("Edit2"), 2))

    DaysCalc

    Text(3).SetWindowText Trim$(Str$(SMM)) & "ヶ月"
    Text(5).SetWindowText Trim$(Str$(SDD)) & "日"
End Sub

'=====================================
'= Edit1
'=====================================
Declare Sub Edit1_Change edecl ()
Sub Edit1_Change()
    ED1$ = GetDlgItemText("Edit1")
    ePos = InStr(ED1$, CrLf)
    If ePos <> 0 Then
        ED1$ = Mid$(ED1$, 1, ePos - 1) & Mid$(ED1$, ePos + 2)
        Edit(0).SetWindowText ED1$
        SY1 = Val(Left$(ED1$, 4))
        SM1 = Val(Mid$(ED1$, 5, 2))
        SD1 = Val(Right$(ED1$, 2))

        SY = SY1
        SM = SM1
        SD = SD1
        EndOfMonth

        If SM1 < 1 Or SM1 > 12 Or SD1 < 1 Then
            Edit(1).SetWindowText ""
            Edit(1).SetFocus
        Else If SD1 > SE Then
            ED1$ = Left$(ED1$, 6) & Right$(Str$(100 + SE), 2)
            Edit(0).SetWindowText ED1$
        End If

        Edit(1).SetWindowText ""
        Edit(1).SetFocus
     end if
End Sub

'=====================================
'= Edit2
'=====================================
Declare Sub Edit2_Change edecl ()
Sub Edit2_Change()
    ED2$ = GetDlgItemText("Edit2")
    ePos = InStr(ED2$, CrLf)

    If ePos <> 0 Then
        ED2$ = Mid$(ED2$, 1, ePos - 1) & Mid$(ED2$, ePos + 2)
        Edit(1).SetWindowText ED2$
        SY2 = Val(Left$(ED2$, 4))
        SM2 = Val(Mid$(ED2$, 5, 2))
        SD2 = Val(Right$(ED2$, 2))

        SY = SY2
        SM = SM2
        SD = SD2
        EndOfMonth

        If SM2 < 1 Or SM2 > 12 Or SD2 < 1 Then
            Edit(1).SetWindowText ""
            Edit(1).SetFocus
        Else If SD2 > SE Then
            ED2$ = Left$(ED2$, 6) & Right$(Str$(100 + SE), 2)
```

```
            Edit(1).SetWindowText ED2$
        End If
        Button1.SetFocus
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

ディスクドライブにメディアが挿入されているか判定(Ⅰ)

GetLogicalDrives 利用可能ディスクドライブ取得

Visual Basic では、IsReady などで、簡単に取得できるのですが…
例では、エラーが発生した場合、メディアが準備されていないと判断しています。




```
'=====================================
'= ディスクドライブにメディアが挿入されているか判定
'=    (MediaInDrive.bas)
'=====================================
#include "Windows.bi"
#include "File.bi"

' 利用可能ディスクドライブ取得
Declare Function Api_GetLogicalDrives& Lib "kernel32" Alias "GetLogicalDrives" ()

Var Shared Combo1 As Object
Var Shared Button1 As Object

Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var Shared Drv As String

'=====================================
'=
'=====================================
Declare Function MediaCheck(DriveLetter As String) As Integer
Function MediaCheck(DriveLetter As String) As Integer
    Var Target As String
    Var Ret As String

    On Error GoTo *Er_Trap

    '「:」が無い場合の処理
    Target = Left$(DriveLetter, 1)
    Target = Target & ":"

    Ret = CrDir$(Target)

    MediaCheck = True

*Er_Return
```

```
    On Error GoTo 0
    Exit Function

*Er_Trap
    MediaCheck = False
    Resume *Er_Return
End Function

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var Ret As Long

    Ret = Api_GetLogicalDrives()            '利用可能なディスクドライブ取得
    If Ret = 0 Then Exit Sub                '関数の失敗

    For i = 0 To 25                         'A～Zドライブを検索する
        If (Ret And 1) = 1 Then
            Drv = Chr$(65 + i)              'ドライブ名(A～Z)に変換
            Drv = Drv & ":¥"
            Combo1.AddString Drv
        End If
        Ret = Ret ¥ 2                       'ドライブ検索
    Next i
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Drv = Combo1.GetWindowText

    If MediaCheck(Drv) = False Then
        A% = MessageBox("", "メディアを挿入してください", 0, 2)
    Else
        A% = MessageBox("", "アクセス可能です。", 0, 2)
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

ディスクドライブにメディアが挿入されているか判定(II)

GetDriveType ドライブのタイプを取得
GetFileAttributes 指定されたファイルまたはディレクトリの属性を取得
SetErrorMode エラーに対するオペレーティングシステムの処理方法を指定
GetLogicalDrives 利用可能ディスクドライブ取得
GetLastError エラーコードを取得

```
'=====================================
'= ディスクドライブにメディアが挿入されているか判定(II)
'=    (MediaInDrive2.bas)
'=====================================
#include "Windows.bi"
#include "File.bi"

#define DRIVE_NODETERMINE_DRIVETYPE 0      '
#define DRIVE_NOEXIST_ROOTDIRECTORY 1      '
#define DRIVE_CDROM 5                      'CD-ROMドライブ
#define DRIVE_FIXED 3                      '固定タイプ－主にハードディスク
#define DRIVE_NO_ROOT_DIR 1                'ルートディレクトリ無し
#define DRIVE_RAMDISK 6                    'RAMドライブ
#define DRIVE_REMOTE 4                     'ネットワーク
#define DRIVE_REMOVABLE 2                  '取り外し可能タイプ－リムーバブルディスク
#define DRIVE_UNKNOWN 0                    'ドライブが不明
#define ERROR_NOT_READY 21                 '
#define ERROR_GEN_FAILURE 31               '
#define SEM_FAILCRITICALERRORS &H1         'OSは致命的なエラーに関するメッセージを表示しない
#define SEM_NOGPFAULTERRORBOX &H2          'OSはメモリ整列違反を自動的に修復する
#define SEM_NOOPENFILEERRORBOX &H800       'OSはファイルが見つからなかった時にメッセージを表示しない
#define FORMAT_MESSAGE_IGNORE_INSERTS &H200      'Arguments パラメータを無視するよう要求
#define FORMAT_MESSAGE_FROM_SYSTEM &H1000 'メッセージ定義として、システムメッセージテーブルリソースを使用するよう要求
#define PROCESS_DEFAULT_LANGUAGE &H400     'デフォルト言語を指定

' ドライブのタイプを取得
Declare Function Api_GetDriveType& Lib "Kernel32" Alias "GetDriveTypeA" (ByVal nDrive$)

' 指定されたファイルまたはディレクトリの属性を取得
Declare Function Api_GetFileAttributes& Lib "Kernel32" Alias "GetFileAttributesA" (ByVal
lpFileName$)

' エラーに対するオペレーティングシステムの処理方法を指定
Declare Function Api_SetErrorMode& Lib "kernel32" Alias "SetErrorMode" (ByVal uMode&)

' 利用可能ディスクドライブ取得
Declare Function Api_GetLogicalDrives& Lib "kernel32" Alias "GetLogicalDrives" ()

' エラーコードを取得
Declare Function Api_GetLastError& Lib "Kernel32" Alias "GetLastError" ()

Var Shared Combo1 As Object
Var Shared Text1 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var Shared Drv As String

'=====================================
'=
'=====================================
Declare Function MediaCheck(DriveLetter As String) As Integer
Function MediaCheck(DriveLetter As String) As Integer
    Var Target As String
    Var Ret As String

    On Error GoTo *Er_Trap

    '「:」が無い場合の処理
    Target = Left$(DriveLetter, 1)
    Target = Target & ":"

    Ret = CrDir$(Target)

    MediaCheck = True
```

```
*Er_Return
    On Error GoTo 0
    Exit Function

*Er_Trap
    MediaCheck = False
    Resume *Er_Return

End Function

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var Ret As Long

    '利用可能なディスクドライブ取得
    Ret = Api_GetLogicalDrives()

    '関数の失敗
    If Ret = 0 Then Exit Sub

    'A～Zドライブを検索する
    For i = 0 To 25
        If (Ret And 1) = 1 Then

            'ドライブ名(A～Z)に変換
            Drv = Chr$(65 + i)
            Drv = Drv & ":¥"
            Combo1.AddString Drv
        End If

        'ドライブ検索
        Ret = Ret ¥ 2
    Next i
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var RootPathName As String
    Var DriveType As Long
    Var Attr As Long
    Var Flag As Long
    Var Ret As Long

    Flag = FORMAT_MESSAGE_FROM_SYSTEM Or FORMAT_MESSAGE_IGNORE_INSERTS

    Button1.EnableWindow 0

    'ルートディレクトリを指定
    RootPathName = Combo1.GetWindowText

    'ドライブの種類を取得
    DriveType = Api_GetDriveType(RootPathName)

    'ドライブがディスク脱着可能であるとき
    If DriveType = DRIVE_REMOVABLE Then

        'システムエラーを抑制
        Ret = Api_SetErrorMode(SEM_FAILCRITICALERRORS)

        'ルートディレクトリのファイル属性を取得
        Attr = Api_GetFileAttributes(RootPathName)

        'ファイル属性を取得できないときは
        If Attr = -1 Then
```

```
            Select Case Api_GetLastError()
                Case ERROR_NOT_READY
                    Text1.SetWindowText "装着されていません。"
                Case ERROR_GEN_FAILURE
                    Text1.SetWindowText "フォーマットされていません。"
                Case Else
                    Text1.SetWindowText "状態を取得できません。"
            End Select
        Else
            Text1.SetWindowText "装着されています。"
        End If

        'システムエラーを既定に設定
        Ret = Api_SetErrorMode(0)
    Else
        Text1.SetWindowText "脱着可能ドライブではありません。"
    End If

    'コマンドボタンを有効に設定
    Button1.EnableWindow -1
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

ディスクドライブのシリアル番号を取得(Ⅰ)

指定したドライブのシリアル番号を取得します。
GetLogicalDrives 利用可能ディスクドライブ取得
GetVolumeInformation ルートディレクトリが呼び出しで指定されたファイルシステムとボリュームについての情報を返す

※一見固有の値のようですが、メーカーパソコンで同一機種のボリュームシリアルが、全て同一ということもあるそうです。MACアドレスとハードディスクボリュームシリアル番号の組み合わせが、絶対ではありませんが確率的には利用可能かも••

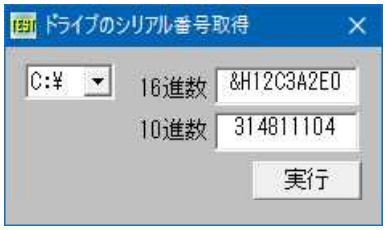
```
'=======================================
'= デスクドライブのシリアル番号取得(Ⅰ)
'=    (GetVolumeInformation.bas)
'=======================================
#include "Windows.bi"

' 利用可能ディスクドライブ取得
Declare Function Api_GetLogicalDrives& Lib "kernel32" Alias "GetLogicalDrives" ()

' ルート ディレクトリが呼び出しで指定されたファイル システムとボリュームについての情報を返す
Declare Function Api_GetVolumeInformation& Lib "kernel32" Alias "GetVolumeInformationA"
(ByVal RootPathName$, ByVal VolNameBuff$, ByVal VolNameSize&, VolSerialNum&,
MaxComponentLen&, FileSysFlag&, ByVal FileSysNameBuff$, ByVal FileSysNameSize&)

#define MAX_PATH 260

Var Shared Text(3) As Object
```

```
Var Shared Combo1 As Object
Var Shared Button1 As Object

For i = 0 To 3
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1)))
    Text(i).SetFontSize 14
Next
Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var Shared Drive As String

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Drives = Api_GetLogicalDrives()          '利用可能なディスクドライブ取得
    If Drives = 0 Then Exit Sub              '関数の失敗

    For i = 0 To 25                          'A～Zドライブを検索する
        If (Drives And 1) = 1 Then
            Drive = Chr$(65 + i)             'ドライブ名(A～Z)に変換
            Drive = Drive & ":¥"
            Combo1.AddString Drive
        End If
        Drives = Drives ¥ 2                  'ドライブ検索
    Next i
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var VolLabel As String
    Var Serial As Long
    Var MaxLen As Long
    Var Flags As Long
    Var nName As String
    Var s As String
    Var Ret As Long

    Drive = Left$(Combo1.GetText(Combo1.GetCursel), 3)

    Text(0).SetWindowText ""
    Text(1).SetWindowText ""

    If Api_GetVolumeInformation(Drive, VolLabel, MAX_PATH, Serial, MaxLen, Flags, nName,
MAX_PATH) Then
        Text(0).SetWindowText "&&H" & Hex$(Serial)
        Text(1).SetWindowText Trim$(Str$(Serial))
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

デスクドライブのシリアル番号取得(II)

指定したドライブのシリアル番号を取得します。

GetLogicalDrives 利用可能ディスクドライブ取得
GetVolumeSerialNumber ドライブのシリアル番号を取得



※一見固有の値のようですが、メーカーパソコンで同一機種のボリュームシリアルが、全て同一ということもあるそうです。
MACアドレスとハードディスクボリュームシリアル番号の組み合わせが、絶対ではありませんが確率的には利用可能かも••

```
'=====================================
'= デスクドライブのシリアル番号取得(II)
'=    (GetVolumeSerialNumber.bas)
'=====================================
#include "Windows.bi"

' 利用可能ディスクドライブ取得
Declare Function Api_GetLogicalDrives& Lib "kernel32" Alias "GetLogicalDrives" ()

'ドライブのシリアル番号を取得
Declare Function Api_GetVolumeSerialNumber& Lib "kernel32" Alias
"GetVolumeInformationA" (ByVal PathName$, ByVal VolNameBuff&, ByVal VolNameSize&,
VolSerialNum&, ByVal MaxComponentLen&, ByVal FileSysFlags&, ByVal FileSysNameBuff&,
ByVal FileSysNameSize&)

#define MAX_PATH 260

Var Shared Text(3) As Object
Var Shared Combo1 As Object
Var Shared Button1 As Object

For i = 0 To 3
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1)))
    Text(i).SetFontSize 14
Next
Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var Shared Drive As String

'=====================================
'=
'=====================================
Declare Function VolumeSerial(DriveLetter As String) As Long
Function VolumeSerial(DriveLetter As String) As Long
    var Serial As Long
    var Ret As Long
    Ret = Api_GetVolumeSerialNumber(Ucase$(DriveLetter), 0, 0, Serial, 0, 0, 0, 0)
    VolumeSerial = Serial
End Function

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Drives = Api_GetLogicalDrives()          '利用可能なディスクドライブ取得
    If Drives = 0 Then Exit Sub              '関数の失敗

    For i = 0 To 25                          'A～Zドライブを検索する
        If (Drives And 1) = 1 Then
            Drive = Chr$(65 + i)             'ドライブ名(A～Z)に変換
            Drive = Drive & ":¥"
            Combo1.AddString Drive
```

```
        End If
        Drives = Drives ¥ 2                    'ドライブ検索
    Next i
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var VolLabel As String
    Var Serial As Long
    Var MaxLen As Long
    Var Flags As Long
    Var nName As String
    Var s As String
    Var Ret As Long

    Drive = Left$(Combo1.GetText(Combo1.GetCursel), 3)

    Text(0).SetWindowText "&&H" & Hex$(VolumeSerial(Drive))
    Text(1).SetWindowText Trim$(Str$(VolumeSerial(Drive)))
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

ディスクドライブのシリアル番号を取得（III）

GetVolumeSerialNumber ドライブのシリアル番号を取得

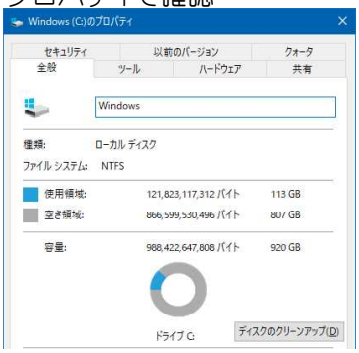

```
'=====================================
'=ドライブのシリアル番号を取得
'=    (GetVolumeSerialNumber.bas)
'=====================================

'ドライブのシリアル番号を取得
Declare Function Api_GetVolumeSerialNumber& Lib "kernel32" Alias
"GetVolumeInformationA" (ByVal PathName$, ByVal VolNameBuff&, ByVal VolNameSize&,
VolSerialNum&, ByVal MaxComponentLen&, ByVal FileSysFlags&, ByVal FileSysNameBuff&,
ByVal FileSysNameSize&)

Declare Function VolumeSerial(DriveLetter As String) As Long
Function VolumeSerial(DriveLetter As String) As Long
    Var Serial As Long
    Var Ret As Long
    Ret = Api_GetVolumeSerialNumber(ucase$(DriveLetter) & ":¥", 0, 0, Serial, 0, 0, 0, 0)
    VolumeSerial = Serial
End Function
```

```
Do
    Input "ドライブ:", DRV$

    Print "&H" & Hex$(VolumeSerial(DRV$)) & "(" & Trim$(Str$(VolumeSerial(DRV$))) & ")"
    Print
Loop

Stop
End
```

ディスクの空き容量を取得(Ⅰ)

ディスクの空き容量を取得します。(Window2000/WindowXP)
GetLogicalDrives 利用可能ディスクドライブ取得
GetDiskFreeSpace 指定のディレクトリを含むディスクの空き容量を調べる

プロパティで確認



プロパティで確認


```
'=======================================
'= ディスクの空き容量を取得(Window2000/WindowXP)
'=    (GetDiskFreeSpace.bas)
'=======================================
#include "Windows.bi"

' 利用可能ディスクドライブ取得
Declare Function Api_GetLogicalDrives& Lib "kernel32" Alias "GetLogicalDrives" ()

' 指定のディレクトリを含むディスクの空き容量を調べる
Declare Function Api_GetDiskFreeSpace& Lib "kernel32" Alias "GetDiskFreeSpaceA" (ByVal
lpRootPathName$, lpSecPerCluster&, lpBytesPerSector&, lpNumberOfFreeCs&,
lpTotalNumberOfClusters&)

Var Shared Combo1 As Object
Var Shared Text(7) As Object
Var Shared Picture1 As Object

Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
Picture1.Attach GetDlgItem("Picture1")
For i = 0 To 7
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1)))
    Text(i).SetFontSize 14
Next
```

```
Var Shared Drive As String

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Drives = Api_GetLogicalDrives()          '利用可能なディスクドライブ取得
    If Drives = 0 Then Exit Sub              '関数の失敗

    For i = 0 To 25                          'A～Zドライブを検索する
        If (Drives And 1) = 1 Then
            Drive = Chr$(65 + i)             'ドライブ名(A～Z)に変換
            Drive = Drive & ":¥"
            Combo1.AddString Drive
        End If
        Drives = Drives ¥ 2                  'ドライブ検索
    Next i
End Sub

'=====================================
'=
'=====================================
Declare Sub Combo1_Change edecl ()
Sub Combo1_Change()
    Var Sector As Long
    Var Bytes As Long
    Var FreeC As Long
    Var TotalC As Long
    Var Ret As Long

    'ドライブ名取得
    Drive = Left$(Combo1.GetText(Combo1.GetCursel), 3)
    Ret = Api_GetDiskFreeSpace(Drive, Sector, Bytes, FreeC, TotalC)

    If RET <> 0 Then

        '総容量 = 総クラスタ数 * クラスタ当たりのセクタ数 * セクタ当たりのバイト数
        Text(4).SetWindowText Format$(TotalC / 1000000 * Sector * Bytes, "#,###,###.##")

        '使用容量 = 総容量 - 空き容量
        Text(5).SetWindowText Format$((TotalC / 1000000 * Sector * Bytes) - (FreeC /
1000000 * Sector * Bytes), "#,###,###.##")

        '空き容量 = 空きクラスタ数 * クラスタ当たりのセクタ数 * セクタ当たりのバイト数
        Text(6).SetWindowText Format$(FREEC / 1000000 * Sector * Bytes, "#,###,###.##")

        '空き容量のパーセンテージ
        Text(7).SetWindowText Format$((FreeC / 1000000 * Sector * Bytes) / (TotalC /
1000000 * Sector * Bytes) * 100, "##.##")

        'グラフ作成
        Picture1.Cls
        Picture1.Connect(0, Picture1.GetHeight - 3) - (Picture1.GetWidth - 3,
Picture1.GetHeight - 3) - (Picture1.Getwidth - 3, 0), 15
        Picture1.Line(2, 1) - ((FreeC / 1000000 * Sector * Bytes) / (TotalC / 1000000 *
Sector * Bytes) * Picture1.GetWidth , Picture1.GetHeight - 4), , 7, bf
        Picture1.Line((FreeC / 1000000 * Sector * Bytes) / (TotalC / 1000000 * Sector *
Bytes) * Picture1.GetWidth + 1, 1) - (Picture1.Getwidth - 4, Picture1.GetHeight - 4), , 3,
bf
    Else
        For i = 4 To 7
            Text(i).SetWindowText ""
        Next
        Picture1.Cls
    End If
End Sub
```

```
'=====================================
'=
'=====================================
Declare Function GetTotalSpace(Drive As String) As Single
Function GetTotalSpace(Drive As String) As Single
    Var SectorPerCluster As Long
    Var BytePerCluster As Long
    Var FreeC As Long
    Var TotalC As Long

    If Api_GetDiskFreeSpace(Drive, SectorPerCluster, BytePerCluster, FreeC, TotalC) = 0
Then
        GetTotalSpace = 0
    Else
        GetTotalSpace = ((TotalC / 1000000) * SectorPerCluster * BytePerCluster)
    End If
End Function

'=====================================
'=
'=====================================
Declare Function GetFreeSpace(Drive As String) As Single
Function GetFreeSpace(Drive As String) As Single
    Var SectorPerCluster As Long
    Var BytePerCluster As Long
    Var FreeC As Long
    Var TotalC As Long

    If Api_GetDiskFreeSpace(Drive, SectorPerCluster, BytePerCluster, FreeC, TotalC) = 0
Then
        GetFreeSpace = 0
    Else
        GetFreeSpace = ((FreeC / 1000000) * SectorPerCluster * BytePerCluster)
    End If
End Function

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

ディスクの空き容量を取得(II)

ディスクの空き容量を取得します。VBでのCurrencyに相当する数値変数がないので、ULARGE_INTEGER構造体を用いてLowPart・HighPartに分けて計算しています。
GetLogicalDrives 使用可能ドライブの取得
GetLogicalDriveStrings 有効なドライブ名の取得
GetDiskFreeSpaceEx 指定のディレクトリを含むディスクの空き容量を調べる

コンボボックスには現在使用できるドライブがセットされています。目的ドライブをクリックすると容量を取得することができます。

Windows XP・2000・98での結果と、それぞれのプロパティでの確認を示しています。

参考

ビット	bit(b)	
バイト	byte(B)	1B = 8b
キロバイト	Kilo Byte(KB)	1KB = 1,000B
メガバイト	Mega Byte(MB)	1MB = 1,000KB
ギガバイト	Giga Byte(GB)	1GB = 1,000MB
テラバイト	Tera Byte(TB)	1TB = 1,000GB
ペタバイト	Peta Byte(PB)	1PB = 1,000TB
エクサバイト	Exa Byte(EB)	1EB = 1,000PB
ゼタバイト	Zetta Byte(ZB)	1ZB = 1,000EB
ヨタバイト	Yotta Byte(YB)	1YB = 1,000ZB

```
'=======================================
'= ディスクの空き容量を取得(II)
'=    (GetDiskFreeSpaceEx2.bas)
'=======================================
#include "Windows.bi"

Type ULARGE_INTEGER
    lowpart    As Long
    highpart   As Long
End Type

' 使用可能ドライブの取得
Declare Function Api_GetLogicalDrives& Lib "kernel32" Alias "GetLogicalDrives" ()

' 有効なドライブ名の取得
Declare Function Api_GetLogicalDriveStrings& Lib "kernel32" Alias
"GetLogicalDriveStringsA" (ByVal nBufferLength&, ByVal lpBuffer$)

' 指定のディレクトリを含むディスクの空き容量を調べる
Declare Function Api_GetDiskFreeSpaceEx& Lib "kernel32" Alias "GetDiskFreeSpaceExA"
(ByVal lpDirectoryName$, lpFreeBytesAvailableToCaller As ULARGE_INTEGER,
lpTotalNumberOfBytes As ULARGE_INTEGER, lpTotalNumberOfFreeBytes As ULARGE_INTEGER)

Var Shared Combo1 As Object
Var Shared Text(9) As Object

Combo1.Attach GetDlgItem("Combo1")
Combo1.SetFontSize 14
For i = 0 To 9
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1)))
    Text(i).SetFontSize 14
Next

'=======================================
'=
'=======================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var Drv As String                          'ドライブ名
```

```
    Var Drives As Long                          'ディスクドライブのビットマスク

    Drives = Api_GetLogicalDrives()             '利用可能なディスクドライブ取得
    If Drives = 0 Then exit Sub                 '関数の失敗

    For i = 0 To 25                             'A～Zドライブを検索する
        If (Drives and 1) = 1 Then
            Drv = Chr$(65 + i)                  'ドライブ名(A～Z)に変換
            Drv = Drv & ":¥"
            Combo1.AddString Drv
        End If
        Drives = Drives ¥ 2                     'ドライブ検索
    Next i
End Sub

'=====================================
'=
'=====================================
Declare Function LargeIntegerToDouble(LowPart As Long, HighPart As Long) As Double
Function LargeIntegerToDouble(LowPart As Long, HighPart As Long) As Double
    Var Ret As Double

    Ret = HighPart
    If HighPart < 0 Then Ret = Ret + 2 ^ 32
    Ret = Ret * 2 ^ 32

    Ret = Ret + LowPart
    If LowPart < 0 Then Ret = Ret + 2 ^ 32

    LargeIntegerToDouble = Ret
End Function

'=====================================
'=
'=====================================
Declare Function ThreeNonZeroDigits(Value As Double) As String
Function ThreeNonZeroDigits(Value As Double) As String
    If Value >= 100 Then

        '小数無し
        ThreeNonZeroDigits = Format$(Int(Value),"### ")
    Else If Value >= 10 Then

        '小数点以下1桁
        ThreeNonZeroDigits = Format$(Value, "###.# ")
    Else

        '小数点以下2桁
        ThreeNonZeroDigits = Format$(Value, "###.##")
    End If
End Function

'=====================================
'=
'=====================================
Declare Function FormatBytes(bNum As Double) As String
Function FormatBytes(bNum As Double) As String
    Static ONE_KB As Double : ONE_KB = 1024
    Static ONE_MB As Double : ONE_MB = ONE_KB * 1024
    Static ONE_GB As Double : ONE_GB = ONE_MB * 1024
    Static ONE_TB As Double : ONE_TB = ONE_GB * 1024
    Static ONE_PB As Double : ONE_PB = ONE_TB * 1024
    Static ONE_EB As Double : ONE_EB = ONE_PB * 1024
    Static ONE_ZB As Double : ONE_ZB = ONE_EB * 1024
    Static ONE_YB As Double : ONE_YB = ONE_ZB * 1024

    Var Value As Double

    If bNum <= 999 Then
```

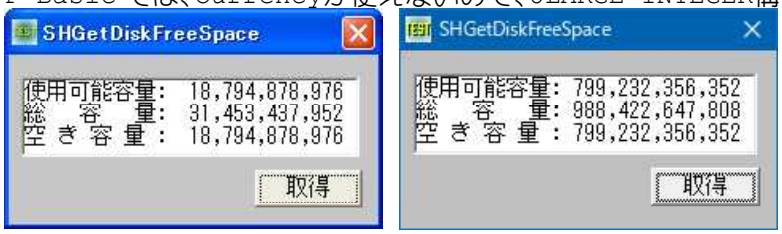
```
        'bytesフォーマット
        FormatBytes = Format$(bNum, "### ") & "B"
    Else If bNum <= ONE_KB * 999 Then

        'KBフォーマット
        FormatBytes = ThreeNonZeroDigits(bNum / ONE_KB) & "KB"
    Else If bNum <= ONE_MB * 999 Then

        'MBフォーマット
        FormatBytes = ThreeNonZeroDigits(bNum / ONE_MB) & "MB"
    Else If bNum <= ONE_GB * 999 Then

        'GBフォーマット
        FormatBytes = ThreeNonZeroDigits(bNum / ONE_GB) & "GB"
    Else If bNum <= ONE_TB * 999 Then

        'TBフォーマット
        FormatBytes = ThreeNonZeroDigits(bNum / ONE_TB) & "TB"
    Else If bNum <= ONE_PB * 999 Then

        'PBフォーマット
        FormatBytes = ThreeNonZeroDigits(bNum / ONE_PB) & "PB"
    Else If bNum <= ONE_EB * 999 Then

        'EBフォーマット
        FormatBytes = ThreeNonZeroDigits(bNum / ONE_EB) & "EB"
    Else If bNum <= ONE_ZB * 999 Then

        'ZBフォーマット
        FormatBytes = ThreeNonZeroDigits(bNum / ONE_ZB) & "ZB"
    Else

        'YBフォーマット
        FormatBytes = ThreeNonZeroDigits(bNum / ONE_YB) & "YB"
    End If
End Function

'=====================================
'=
'=====================================
Declare Sub Combo1_Change edecl ()
Sub Combo1_Change()
    Var bAvail As ULARGE_INTEGER
    Var bTotal As ULARGE_INTEGER
    Var bFree As ULARGE_INTEGER
    Var dTotal As Double
    Var dFree As Double
    Var Ret As Long

    Ret = Api_GetDiskFreeSpaceEx(GetDlgItemText("Combo1"), bAvail, bTotal, bFree)

    dTotal = LargeIntegerToDouble(bTotal.lowpart, bTotal.highpart)

    If dTotal = 0 Then
        For i = 0 To 5 : Text(i).SetWindowText "" : Next
            Res = MessageBox("", "取得できません！", 0, 2)
            Exit Sub
    End If

    dFree = LargeIntegerToDouble(bFree.lowpart, bFree.highpart)

    Text(0).SetWindowText FormatBytes(dTotal)
    Text(1).SetWindowText FormatBytes(dFree)
    Text(2).SetWindowText FormatBytes(dTotal - dFree)
    Text(3).SetWindowText Format$(100, "###.##%")
    Text(4).SetWindowText Format$(dFree / dTotal * 100, "###.##%")
    Text(5).SetWindowText Format$((dTotal - dFree) / dTotal * 100, "###.##%")
End Sub
```

```
'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

ディスクの空き容量を取得(III)

SHGetDiskFreeSpace ディスクの空き容量を調べる

F-Basic では、Currencyが使えないので、ULARGE_INTEGER構造体でHigh、Lowに分けて取り出しています。

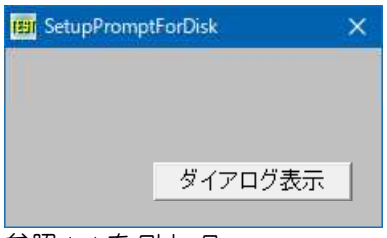


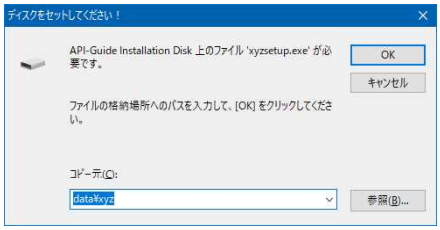

```
'=====================================
'= ディスクの空き容量を取得(III)
'=    (SHGetDiskFreeSpace.bas)
'=====================================
#include "Windows.bi"

Type ULARGE_INTEGER
    LowPart    As Long
    HighPart   As Long
End Type

' 指定のディレクトリを含むディスクの空き容量を調べる
Declare Function Api_SHGetDiskFreeSpace& Lib "Shell32" Alias "SHGetDiskFreeSpaceA"
(ByVal pszVolume$, pqwFreeCaller As ULARGE_INTEGER, pqwTot As ULARGE_INTEGER, pqwFree As
ULARGE_INTEGER)

#define vbCrLf (Chr$(13) & Chr$(10))        'キャリッジリターンとラインフィード(¥r¥n)

Var Shared Text1 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Function ULARGE_INTEGER_DOUBLE(ui As ULARGE_INTEGER) As Double
Function ULARGE_INTEGER_DOUBLE(ui As ULARGE_INTEGER) As Double
    Var lHigh As Long
    Var lLow As Long

    lHigh = ui.HighPart * 2
    lLow = ui.LowPart

    If lLow And (-2147483648) THen
        lHigh = lHigh + 1
    End If

    lLow = lLow And &H7FFFFFFF
    ULARGE_INTEGER_DOUBLE = lLow + lHigh * 2 ^ 31
End Function
```

```
'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Drive As String
    Var FreeCaller As ULARGE_INTEGER
    Var Total As ULARGE_INTEGER
    Var Free As ULARGE_INTEGER
    Var txt As String
    Var Ret As Long
    Drive = "C:¥"

    Ret = Api_SHGetDiskFreeSpace(Drive, FreeCaller, Total, Free)

    txt = txt & "使用可能容量: " & Format$(ULARGE_INTEGER_DOUBLE(FreeCaller),
"###,###,###,###") & vbCrLf
    txt = txt & "総　容　量: " & Format$(ULARGE_INTEGER_DOUBLE(Total)    ,
"###,###,###,###") & vbCrLf
    txt = txt & "空 き 容 量 : " & Format$(ULARGE_INTEGER_DOUBLE(Free)     ,
"###,###,###,###")
    Text1.SetWindowText txt
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

ディスクの交換を求めるダイアログボックスを表示

SetupPromptForDisk ユーザーにディスクの交換を求めるダイアログボックスを表示





参照(B)をクリック

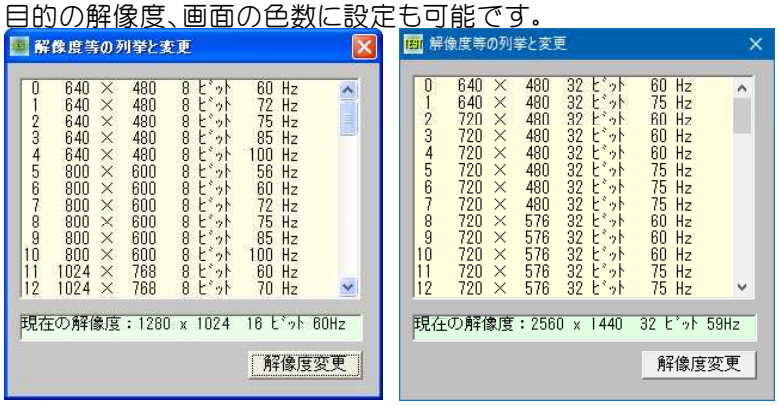

```
'=====================================
'= ディスクの交換を求めるダイアログボックスを表示
'=    (SetupPromptForDisk.bas)
'=====================================
#include "Windows.bi"

'ユーザーにディスクの交換を求めるダイアログボックスを表示
Declare Function Api_SetupPromptForDisk& Lib "setupapi" Alias "SetupPromptForDiskA"
```

```
(ByVal hParent&, ByVal dTitle$, ByVal dName$, ByVal PathToSource$, ByVal fSought$, ByVal
tFile$, ByVal dPromptStyle&, ByVal pBuff$, ByVal pBuffSize&, ByRef pRequiredSize&)

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    Ret = Api_SetupPromptForDisk(GethWnd, "ディスクをセットしてください！", "API-Guide
Installation Disk", "data¥xyz", "xyzsetup.exe", "API-Guide Setup Executable", 0, ByVal 0,
0, ByVal 0)

End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

ディスプレイ解像度等の表示と変更

表示できる解像度、画面の色を取得しリストに表示します。
EnumDisplaySettings 解像度等の取得
ChangeDisplaySettings 解像度等の変更(Windows3.5(1)以降、Windows95以降)

目的の解像度、画面の色数に設定も可能です。



参考(Windows98、Windows2000以降)

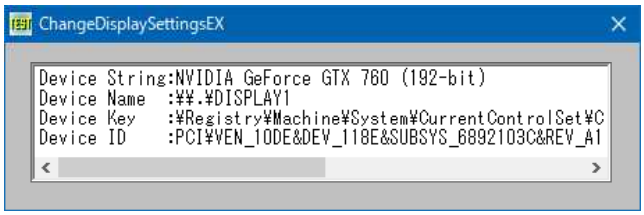
```
Declare Function ChangeDisplaySettingsEx& Lib "user32" Alias "ChangeDisplaySettingsExA"
(lpszDeviceName As Any, lpDevMode As Any, ByVal hWnd&, ByVal dwFlags&, lParam As Any)

'=========================================
'= 解像度変更
'=    (Disp_Setting.bas)
'=========================================
#include "Windows.bi"

#define CCHDEVICENAME 32                    'デバイス名の長さを示す定数
#define CCHFORMNAME 32                      'フォーム名の長さを示す定数
#define ENUM_CURRENT_SETTINGS -1            'ディスプレイ解像度の現在の設定値を求める
#define DM_BITSPERPEL &H40000               '
#define DM_PELSWIDTH &H80000                '
#define DM_PELSHEIGHT &H100000              '

Type DEVMODE
    dmDeviceName      As String * 32        'ドライバがサポートするデバイス名
    dmSpecVersion     As Integer            '構造体の基準になった初期化データ仕様のバージョン番号
    dmDriverVersion   As Integer            'プリンタドライバのバージョン番号
    dmSize            As Integer            'この構造体のサイズ(バイト単位)
```

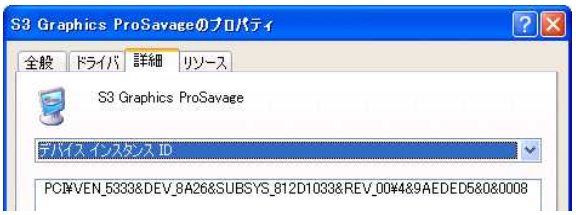
```
    dmDriverExtra     As Integer          'この構造体に続くドライバ データのバイト数
    dmFields          As Long             '
    dmOrientatio      As Integer          'DMORIENT_PORTRAIT、DMORIENT_LANDSCAPE
    dmPaperSize       As Integer          '用紙サイズ
    dmPaperLength     As Integer          'dmPaperSizeメンバで指定した用紙の長さをｵｰﾊﾞｰﾗｲﾄﾞ
    dmPaperWidth      As Integer          'dmPaperSizeメンバで指定した用紙の幅をｵｰﾊﾞｰﾗｲﾄﾞ
    dmScale           As Integer          '印刷出力をスケーリングするときの、スケーリング係数
    dmCopies          As Integer          'デバイスが複数の部数に対応する場合、印刷する部数
    dmDefaultSource   As Integer          '予約済み(0)
    dmPrintQuality    As Integer          'プリンタの解像度(ドット/インチ)
    dmColor           As Integer          'カラープリンタの場合
                                          (DMCOLOR_COLOR･DMCOLOR_MONOCHROME)
    dmDuplex          As Integer          '両面印刷が可能なプリンタ
                                          (DMDUP_SIMPLEX･DMDUP_HORIZONTAL･
                                          DMDUP_VERTICAL)
    dmYResolution     As Integer          'プリンタのy方向の解像度(ドット/インチ)
    dmTTOption        As Integer          'TrueTypeフォントの印刷方法
    dmCollate         As Integer          '複数部数を印刷するときにページ順にそろえるかどうか
    dmFormName        As String * 32      'フォーム名を指定
    dmUnusedPadding   As Integer          '使用しない
    dmBitsPerPixel    As Long             'ディスプレイ デバイスの解像度をピクセルあたりのビット数
                                          で指定
    dmPelsWidth       As Long             '可視のデバイスの表面の幅をピクセル単位で指定
    dmPelsHeight      As Long             '可視のデバイスの表面の高さをピクセル単位で指定
    dmDisplayFlags    As Long             'デバイスのディスプレイ モードを指定
    dmDisplayFrequency As Long            'ディスプレイデバイスのリフレッシュレート(垂直同期周波
                                          数)を1秒当たりのサイクル数(Hz)で指定
End Type

' ディスプレイデバイスのいずれかのグラフィックスモードに関する情報を取得
Declare Function Api_EnumDisplaySettings& Lib "user32" Alias "EnumDisplaySettingsA"
(ByVal lpszDeviceName$, ByVal dwModeNum&,lpDevMode As DEVMODE)

' ディスプレイの解像度を変更
Declare Function Api_ChangeDisplaySettings& Lib "user32" Alias "ChangeDisplaySettingsA"
(lpDevMode As DEVMODE, ByVal dwFlags&)

Var shared List1 As Object
Var shared Text1 As Object
Var shared Button1 As Object

List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var shared  dm As DEVMODE
Var shared  ColorBit As Long
Var shared  DspWidth As Long
Var shared  DspHeight As Long
Var shared  LngNow As Long

'=====================================
'=
'=====================================
Declare Sub Info_Get edecl ()
Sub Info_Get()
    LngNow = Api_EnumDisplaySettings&(ByVal 0 , ENUM_CURRENT_SETTINGS , dm)

    If dm.dmFields And DM_BITSPERPEL Then ColorBit = dm.dmBitsPerPixel
    If dm.dmFields And (DM_PELSWIDTH Or DM_PELSHEIGHT) Then DspWidth = dm.dmPelsWidth :
DspHeight = dm.dmPelsHeight

    Text1.SetWindowText "現在の解像度:" & Trim$(str$(DspWidth)) & " x " &
Trim$(str$(DspHeight)) & " " & str$(ColorBit) & "ﾋﾞｯﾄ" & str$(dm.dmDisplayFrequency) &
"Hz"
End Sub

'=====================================
'=
'=====================================
```

```
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var i As Integer

    For i = 0 To 64
        If Api_EnumDisplaySettings(ByVal 0, i, dm) = 1 Then
            ZD1$ = Format$(i, "##  ")
            ZD2$ = Format$(dm.dmPelsWidth, "#### × ")
            ZD3$ = Format$(dm.dmPelsHeight, "####  ")
            ZD4$ = Format$(dm.dmBitsPerPixel, "## ﾋﾞｯﾄ  ")
            ZD5$ = Format$(dm.dmDisplayFrequency, "### Hz")
            List1.ADDSTRING ZD1$ & ZD2$ & ZD3$ & ZD4$ & ZD5$
        End If
    Next

    Info_Get
End Sub

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    If Api_EnumDisplaySettings(ByVal 0, List1.GetCursel, dm) = 1 Then
        RC = Api_ChangeDisplaySettings(dm, 0)
    End If

    Info_Get
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

ディスプレイ解像度を変更

起動時現在の画面解像度を取得しておき、640×480の解像度に変更します。5秒後解像度を元に戻します。
ついでに、デバイス名等を取得し表示します。
ChangeDisplaySettingsEX 解像度の変更
EnumDisplayDevices ディスプレイに関する情報を取得






```
'=======================================
'= 解像度変更
'=    (ChangeDisplaySettingsEx.bas)
'=======================================
#include "Windows.bi"

#define CCDEVICENAME 32                    'デバイス名の長さを示す定数
#define CCFORMNAME 32                      'フォーム名の長さを示す定数
#define CDS_TEST &H4                       'テストモードにする
Type DISPLAY_DEVICE
    cb                  As Long
    DeviceName          As String * 32
```

```
    DeviceString     As String * 128
    StateFlags       As Long
    DeviceID         As String * 128
    DeviceKey        As String * 128
End Type

Type DEVMODE
    dmDeviceName     As String * CCDEVICENAME
    dmSpecVersion    As Integer
    dmDriverVersion  As Integer
    dmSize           As Integer
    dmDriverExtra    As Integer
    dmFields         As Long
    dmOrientation    As Integer
    dmPaperSize      As Integer
    dmPaperLength    As Integer
    dmPaperWidth     As Integer
    dmScale          As Integer
    dmCopies         As Integer
    dmDefaultSource  As Integer
    dmPrintQuality   As Integer
    dmColor          As Integer
    dmDuplex         As Integer
    dmYResolution    As Integer
    dmTTOption       As Integer
    dmCollate        As Integer
    dmFormName       As String * CCFORMNAME
    dmUnusedPadding  As Integer
    dmBitsPerPel     As Long
    dmPelsWidth      As Long
    dmPelsHeight     As Long
    dmDisplayFlags   As Long
    dmDisplayFrequency As Long
    dmICMMethod      As Long              'NT 4.0
    dmICMIntent      As Long              'NT 4.0
    dmMediaType      As Long              'NT 4.0
    dmDitherType     As Long              'NT 4.0
    dmReserved1      As Long              'NT 4.0
    dmReserved2      As Long              'NT 4.0
    dmPanningWidth   As Long              'Win2000
    dmPanningHeight  As Long              'Win2000
End Type

' ディスプレイの設定を変更
Declare Function Api_ChangeDisplaySettingsEx& Lib "user32" Alias
"ChangeDisplaySettingsExA" (lpszDeviceName As Any, lpDevMode As Any, ByVal hWnd&, ByVal
dwFlags&, lParam As Any)

' ディスプレイに関する情報を取得
Declare Function Api_EnumDisplayDevices& Lib "user32" Alias "EnumDisplayDevicesA"
(Unused As Any, ByVal iDevNum&, lpDisplayDevice As DISPLAY_DEVICE, ByVal dwFlags&)

#define ENUM_CURRENT_SETTINGS -1          'ディスプレイ解像度の現在の設定値を求める
#define DM_BITSPERPEL &H40000             '
#define DM_PELSWIDTH &H80000              '
#define DM_PELSHEIGHT &H100000            '
#define vbCrLf Chr$(13, 10)

Var Shared Edit1 As object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14

Var Shared OldX As Long
Var Shared OldY As Long

'====================================
'=
'====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
```

```
    Var DD As DISPLAY_DEVICE
    Var DM As DEVMODE
    Var Ret As Long

    DD.cb = Len(DD)

    '解像度を取得しておく
    OldX = GetDeviceCaps(8)
    OldY = GetDeviceCaps(10)

    Ret = Api_EnumDisplayDevices(ByVal 0&, 0, DD, ByVal 0&)

    'ディスプレイに関する情報があるとき
    If Ret <> 0 Then
        txt$ = txt$ & "Device String:" & Left$(DD.DeviceString, InStr(1, DD.DeviceString,
Chr$(0)) - 1) & vbCrLf
        txt$ = txt$ & "Device Name  :" & Left$(DD.DeviceName, InStr(1, DD.DeviceName, Chr$
(0)) - 1) & vbCrLf
        txt$ = txt$ & "Device Key   :" & Left$(DD.DeviceKey, InStr(1, DD.DeviceKey,
Chr$(0)) - 1) & vbCrLf
        txt$ = txt$ & "Device ID    :" & Left$(DD.DeviceID, InStr(1, DD.DeviceID, Chr$(0)) -
1)
        Edit1.SetWindowText txt$
    Else
        Edit1.SetWindowText "エラー"
    End If

    DM.dmSize = Len(DM)

    DM.dmFields = DM_PELSWIDTH Or DM_PELSHEIGHT
    DM.dmPelsWidth = 640
    DM.dmPelsHeight = 480

    '解像度を640x480に設定
    Ret = Api_ChangeDisplaySettingsEx(ByVal 0&, DM, ByVal 0&, CDS_TEST, ByVal 0&)

    '5秒後
    Wait 500
    DM.dmPelsWidth = OldX
    DM.dmPelsHeight = OldY

    '起動時の解像度に設定
    Ret = Api_ChangeDisplaySettingsEx(ByVal 0&, DM, ByVal 0&, CDS_TEST, ByVal 0&)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

ディスプレイ上の指定点座標を含むハンドルを取得

MonitorFromPoint 指定の点座標を含むディスプレイモニターのハンドルを取得
ClientToScreen 指定されたウィンドウ上の点の座標を、クライアント領域の座標からスクリーン座標に変換

例では、モニタ座標(0,0)を指定しています。

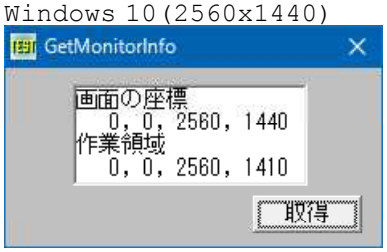

```
'=====================================
'= ディスプレイ上の指定点座標を含むハンドル取得
'=    (MonitorFromPoint.bas)
'=====================================
#include "Windows.bi"

#define MONITOR_DEFAULTTONULL &H0          '0を返す
#define MONITOR_DEFAULTTOPRIMARY &H1       '主モニターのハンドルを返す
#define MONITOR_DEFAULTTONEAREST &H2       '最も近いモニターのハンドルを返す

Type POINTAPI
    x   As Long
    y   As Long
End Type

' 指定の点座標を含むディスプレイモニターのハンドルを取得
Declare Function Api_MonitorFromPoint& Lib "user32" Alias "MonitorFromPoint" (ByVal x&,
ByVal y&, ByVal dwFlags&)

' 指定されたウィンドウ上の点の座標を、クライアント領域の座標からスクリーン座標に変換
Declare Function Api_ClientToScreen& Lib "user32" Alias "ClientToScreen" (ByVal hWnd&,
lpPoint As POINTAPI)

Var Shared Text1 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

ShowWindow -1

'=====================================
'=
'=====================================
Declare Function GetMonitorByPoint(x As Single, y As Single) As Long
Function GetMonitorByPoint(x As Single, y As Single) As Long
    Var pt As POINTAPI
    Var Ret As Long

    pt.x = x
    pt.y = y

    Ret = Api_ClientToScreen(GethWnd, pt)
    GetMonitorByPoint = Api_MonitorFromPoint(pt.x, pt.y, MONITOR_DEFAULTTONEAREST)
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hMonitor As Long
    Var x As Single
    Var y As Single

    x = 0
    y = 0

    hMonitor = GetMonitorByPoint(x, y)
    Text1.SetWindowText "Monitor Handle = &&H" & Hex$(hMonitor)
End Sub
```

```
'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

ディスプレイの座標と作業領域

ディスプレイモニターの画面座標と、作業領域を取得します。
MonitorFromWindow ディスプレイモニターのハンドルを取得
GetMonitorInfo ディスプレイモニターに関する情報を取得(Windows98以降)

Windows 10(2560x1440)



WindowsXP(1280x1024)

Windows2000(1024x768)


```
'=====================================
'= ディスプレイの座標と作業領域
'=    (GetMonitorInfo.bas)
'=====================================
#include "Windows.bi"

#define MONITOR_DEFAULTTONEAREST &H2        '指定したウィンドウに最も近い位置にあるディスプレイモニタのハンドルが返る

#define MONITOR_DEFAULTTONULL &H0           'NULLが返る
#define MONITOR_DEFAULTTOPRIMARY &H1        'プライマリディスプレイモニタのハンドルが返る
#define vbCrLf (Chr$(13) & Chr$(10))        'キャリッジリターンとラインフィード(¥r¥n)
Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

Type MONITORINFO
    cbSize      As Long
    rcMonitor   As RECT
    rcWork      As RECT
    dwFlags     As Long
End Type

' 指定されたウィンドウに外接する四角形との交差部分が最も大きいディスプレイモニタへのハンドルを取得
Declare Function Api_MonitorFromWindow& Lib "user32" Alias "MonitorFromWindow" (ByVal
hWnd&, ByVal dwFlags&)

' ディスプレイモニターに関する情報を取得
Declare Function Api_GetMonitorInfo& Lib "user32" Alias "GetMonitorInfoA" (ByVal
hMonitor&, ByRef lpmi As MONITORINFO)

Var Shared Text1 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
```

```
Sub Button1_on()
    Var mi As MONITORINFO
    Var hMonitor As Long
    Var msg As String

    'モニタのハンドルを取得
    hMonitor = Api_MonitorFromWindow(GethWnd, MONITOR_DEFAULTTONEAREST)
    mi.cbSize = Len(mi)

    'モニタの情報を取得
    If Api_GetMonitorInfo(hMonitor, mi) Then

        '画面の座標
        msg = "画面の座標" & vbCrLf
        msg = msg & "  " & Str$(mi.rcMonitor.Left) & "," & Str$(mi.rcMonitor.Top) & "," &
Str$(mi.rcMonitor.Right) & "," &  Str$(mi.rcMonitor.Bottom) & vbCrLf

        '作業領域の座標
        msg = msg & "作業領域" & vbCrLf
        msg = msg & "  " & Str$(mi.rcWork.Left) & "," & STr$(mi.rcWork.Top) & "," &
Str$(mi.rcWork.Right) & "," &  Str$(mi.rcWork.Bottom)
        Text1.SetWindowtEXT msg
    Else
        Text1.SetWindowText "取得失敗しました！"
    End If
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

ディスプレイモニターに関する情報を取得

EnumDisplayDevices ディスプレイに関する情報を取得
EnumDisplaySettingsEx ディスプレイデバイスのいずれかのグラフィックスモードに関する情報を取得
MonitorFromPoint 指定された点を含むディスプレイモニタへのハンドルを取得
GetMonitorInfo ディスプレイモニターに関する情報を取得


```
'=======================================
'= ディスプレイモニターに関する情報を取得
'=    (GetMonitorInfo2.bas)
'=======================================
#include "Windows.bi"

#define DISPLAY_DEVICE_ATTACHED_TO_DESKTOP &H1
#define DISPLAY_DEVICE_DISCONNECT &H2000000
#define DISPLAY_DEVICE_MIRRORING_DRIVER &H8
#define DISPLAY_DEVICE_MODESPRUNED &H8000000
#define DISPLAY_DEVICE_MULTI_DRIVER &H2
#define DISPLAY_DEVICE_PRIMARY_DEVICE &H4
```

```
#define DISPLAY_DEVICE_REMOTE &H4000000
#define DISPLAY_DEVICE_REMOVABLE &H20
#define DISPLAY_DEVICE_VGA_COMPATIBLE &H10

#define DISPLAY_DEVICE_ACTIVE &H1
#define DISPLAY_DEVICE_ATTACHED &H2

#define CCHDEVICENAME 32                    'デバイス名の長さを示す定数
#define CCHFORMNAME 32                      'フォーム名の長さを示す定数

#define ENUM_CURRENT_SETTINGS -1            'ディスプレイ解像度の現在の設定値を求める
#define ENUM_REGISTRY_SETTINGS -2           '指定した表示デバイスのレジストリに格納されている設定を
                                            取得
#define MONITOR_DEFAULTTONEAREST &H2        '指定したウィンドウに最も近い位置にあるディスプレイモニ
                                            タのハンドルが返る
#define MONITOR_DEFAULTTONULL &H0           'NULLが返る
#define MONITOR_DEFAULTTOPRIMARY &H1        'プライマリディスプレイモニタのハンドルが返る
Type DISPLAY_DEVICE
    cb                  As Long
    DeviceName          As String * 32
    DeviceString        As String * 128
    StateFlags          As Long
    DeviceID            As String * 128
    DeviceKey           As String * 128
End Type

Type POINTAPI
    x   As Long
    y   As Long
End Type

Type DEVMODE
    dmDeviceName            As String * CCHDEVICENAME
    dmSpecVersion           As Integer
    dmDriverVersion         As Integer
    dmSize                  As Integer
    dmDriverExtra           As Integer
    dmFields                As Long
    dmPosition              As POINTAPI
    dmScale                 As Integer
    dmCopies                As Integer
    dmDefaultSource         As Integer
    dmPrintQuality          As Integer
    dmColor                 As Integer
    dmDuplex                As Integer
    dmYResolution           As Integer
    dmTTOption              As Integer
    dmCollate               As Integer
    dmFormName              As String * CCHFORMNAME
    dmLogPixels             As Integer
    dmBitsPerPel            As Long
    dmPelsWidth             As Long
    dmPelsHeight            As Long
    dmDisplayFlags          As Long
    dmDisplayFrequency      As Long
End Type

Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

Type MONITORINFO
    cbSize      As Long
    rcMonitor   As RECT
    rcWork      As RECT
    dwFlags     As Long
End Type
```

```
' ディスプレイに関する情報を取得
Declare Function Api_EnumDisplayDevices& Lib "user32" Alias "EnumDisplayDevicesA"
(Unused As Any, ByVal iDevNum&, lpDisplayDevice As DISPLAY_DEVICE, ByVal dwFlags&)

' ディスプレイデバイスのいずれかのグラフィックスモードに関する情報を取得
Declare Function Api_EnumDisplaySettingsEx& Lib "user32" Alias "EnumDisplaySettingsExA"
(ByVal lpszDeviceName$, ByVal iModeNum&, ByRef lpDevMode As DEVMODE, ByVal dwFlags&)

' 指定された点を含むディスプレイモニタへのハンドルを取得
Declare Function Api_MonitorFromPoint& Lib "user32" Alias "MonitorFromPoint" (ByVal x&,
ByVal y&, ByVal dwFlags&)

' ディスプレイモニターに関する情報を取得
Declare Function Api_GetMonitorInfo& Lib "user32" Alias "GetMonitorInfoA" (ByVal
hMonitor&, ByRef lpmi As MONITORINFO)

Var Shared List1 As Object

List1.Attach GetDlgItem("List1") : List1.SetFontSize 12
List1.SetWindowSize 240, 136

'=======================================
'=
'=======================================
Declare Function CStrToVBStr(str As String) As String
Function CStrToVBStr(str As String) As String
    Var char As String
    Var i As Long
    Var StrRet As String

    For i = 1 To Len(str)
        char = Mid$(str, i, 1)
        If char <> Chr$(0) Then
            StrRet = StrRet & char
        End If
    Next

    CStrToVBStr = StrRet
End Function

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var txt As String
    Var dm As DEVMODE
    Var ddMon As DISPLAY_DEVICE
    Var dd As DISPLAY_DEVICE
    Var mi As MONITORINFO
    Var dev As Long
    Var id As Long
    Var devMon As Long
    Var hm As Long
    Var buf As String
    Var Ret As Long

    txt = ""

    dd.cb = Len(dd)
    dev = 0
    id = 1

    List1.ResetContent

    Do While Api_EnumDisplayDevices(ByVal 0, dev, dd, 0) <> 0
        If Not CInt(dd.StateFlags And DISPLAY_DEVICE_MIRRORING_DRIVER) <> 0 Then
            ddMon.cb = Len(ddMon)
            devMon = 0
```

```
            Do While Api_EnumDisplayDevices(dd.DeviceName, devMon, ddMon, 0) <> 0
                If CInt(ddMon.StateFlags And DISPLAY_DEVICE_ACTIVE) <> 0 Then Exit Do
                devMon = devMon + 1
            Loop

            If CStrToVBStr(ddMon.DeviceString) = "" Then
                Ret = Api_EnumDisplayDevices(dd.DeviceName, 0, ddMon, 0)
                If CStrToVBStr(ddMon.DeviceString) = "" Then ddMon.DeviceString = "Default
Monitor"
            End If

            dm.dmSize = Len(dm)
            If Api_EnumDisplaySettingsEx(dd.DeviceName, ENUM_CURRENT_SETTINGS, dm, 0) = 0
Then
                Ret = Api_EnumDisplaySettingsEx(dd.DeviceName, ENUM_REGISTRY_SETTINGS,
dm, 0)
            End If

            mi.cbSize = Len(mi)
            If CInt(dd.StateFlags And DISPLAY_DEVICE_ATTACHED_TO_DESKTOP) <> 0 Then
                hm = Api_MonitorFromPoint(dm.dmPosition.x, dm.dmPosition.y,
MONITOR_DEFAULTTONULL)
                If hm <> 0 Then
                    Ret = Api_GetMonitorInfo(hm, mi)
                End If
            End If

            List1.AddString "Monitor No." & Str$(id)
            List1.AddString CStrToVBStr(ddMon.DeviceString)
            List1.AddString "(" & CStrToVBStr(dd.DeviceString) & ")"

            If Not CInt(dd.StateFlags And DISPLAY_DEVICE_ATTACHED_TO_DESKTOP) <> 0 Then
                buf = "disabled, "
            Else If CInt(dd.StateFlags And DISPLAY_DEVICE_PRIMARY_DEVICE) <> 0 Then
                buf = "primary, "
            End If

            If CInt(dd.StateFlags And DISPLAY_DEVICE_REMOVABLE) <> 0 Then
                buf = buf & "removable, "
            End If

            If buf <> "" Then
                List1.AddString Left$(buf, Len(buf) - 2)
                buf = ""
            End If

            '幅 x 高さ @ x,y - bpp - リフレッシュレート
            List1.AddString "Width x Height:" & Str$(dm.dmPelsWidth) & " x " &
Str$(dm.dmPelsHeight)
            List1.AddString "Position     :" & Str$(dm.dmPosition.x) & "," &
Str$(dm.dmPosition.y)
            List1.AddString "Bits per Pixel:" & Str$(dm.dmBitsPerPel) & "bit"
            List1.AddString "Refresh Rate  :" & Str$(dm.dmDisplayFrequency) & "Hz"

            'ワークエリア・モニターハンドル
            If hm <> 0 Then
                List1.AddString "WorkSpace    :" & Str$(mi.rcWork.Left) & "," &
Str$(mi.rcWork.Top) & " - " & Str$(mi.rcWork.Right) & "," & Str$(mi.rcWork.Bottom)
                List1.AddString "hMonitor     : &H" & Hex$(hm)
            End If

            'デバイス名
            If ddMon.DeviceName <> "" Then
                List1.AddString CStrToVBStr(ddMon.DeviceName)
            Else
                List1.AddString CStrToVBStr(dd.DeviceName)
            End If
            id = id + 1
        End If
```

```
        dev = dev + 1
    Loop
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

ディレクトリ取得（3題）

ディレクトリ(3)を取得します。
GetCurrentDirectory 自プロセスのカレントディレクトリを取得
GetSystemDirectory Windowsのシステムディレクトリのパスを取得
GetWindowsDirectory Windowsディレクトリのパス名を取得

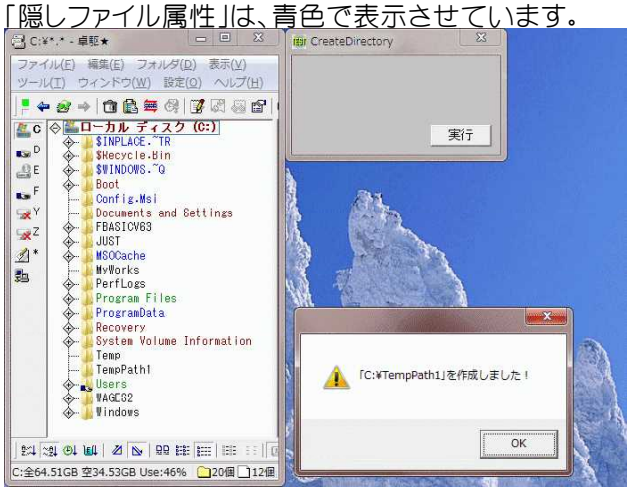

```
'=====================================
'= ディレクトリ取得3題
'=====================================
#include "Windows.bi"

' 自プロセスのカレントディレクトリを取得
Declare Function Api_GetCurrentDirectory& Lib "kernel32" Alias "GetCurrentDirectoryA"
(ByVal nBufferLength&, ByVal lpBuffer$)

' Windows のシステムディレクトリのパスを取得。システムディレクトリには、Windows ライブラリ、ドライバなどのファイルが置かれている
Declare Function Api_GetSystemDirectory& Lib "kernel32" Alias "GetSystemDirectoryA"
(ByVal lpBuffer$, ByVal nSize&)

' Windowsディレクトリのパス名を取得
Declare Function Api_GetWindowsDirectory& Lib "kernel32" Alias "GetWindowsDirectoryA"
(ByVal lpBuffer$, ByVal nSize&)

Var Shared Text1 As Object
Var Shared Text2 As Object
Text1.Attach GetDlgItem("Text1")
Text2.Attach GetDlgItem("Text2")

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Directory As String
    Var Ret As Long

    Directory = String$(255, 0)
    Ret = Api_GetCurrentDirectory(255, Directory)
    Text1.SetWindowText "カレントディレクトリ"
    Text2.SetWindowText Directory
End Sub
```

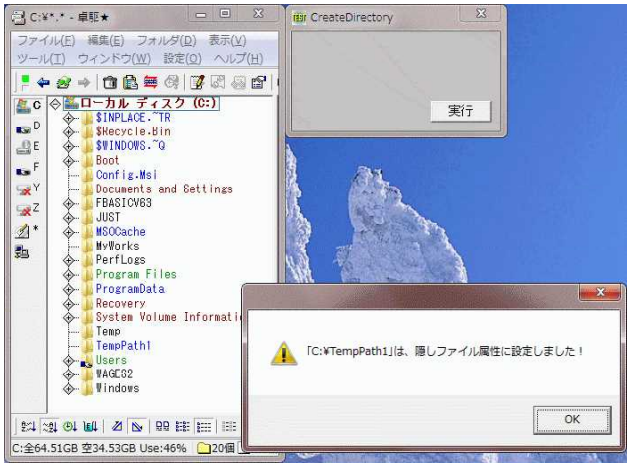
```
'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Directory As String
    Var Ret As Long

    Directory = String$(255, 0)
    Ret = Api_GetSystemDirectory(Directory, 255)
    Directory = Left$(Directory, Ret)
    Text1.SetWindowText "システムディレクトリ"
    Text2.SetWindowText Directory
End Sub

'=====================================
'=
'=====================================
Declare Sub Button3_on edecl ()
Sub Button3_on()
    Var Path As String
    Var Directory As String
    Var Ret As Long

    Directory = String$(255, Chr$(0))
    Ret = Api_GetWindowsDirectory(Directory, Len(Directory))
    Directory = Left$(Directory, Ret)
    Text1.SetWindowText "Windowsディレクトリ"
    Text2.SetWindowText Directory
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

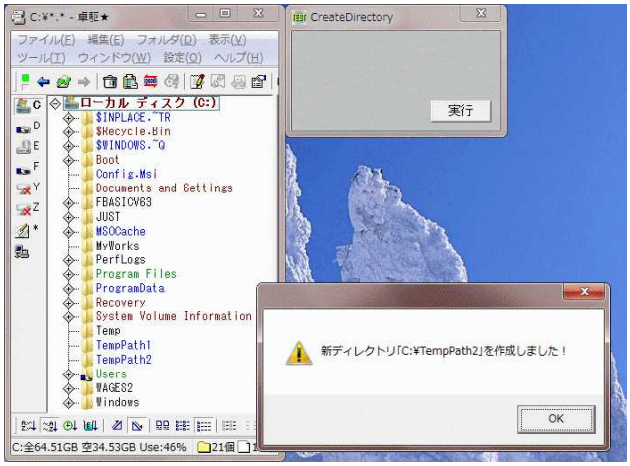
ディレクトリの作成と削除

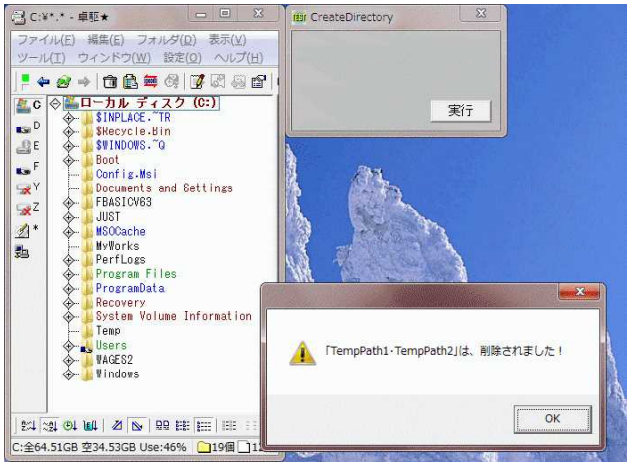
CreateDirectory ディレクトリの新規作成
CreateDirectoryEx ディレクトリの新規作成
RemoveDirectory ディレクトリを削除
SetFileAttributes ファイルの属性を変更

「隠しファイル属性」は、青色で表示させています。







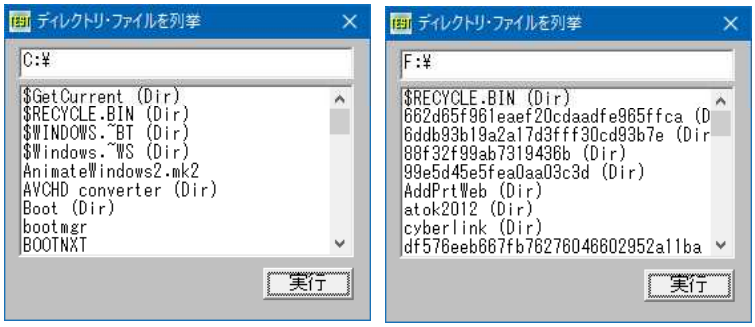



```
'=======================================
'= ディレクトリの作成と削除
'=    (CreateDirectory.bas)
'=======================================
#include "Windows.bi"
Type SECURITY_ATTRIBUTES
    nLength                 As Long
    lpSecurityDescriptor    As Long
    bInheritHandle          As Long
End Type

' ディレクトリの新規作成
Declare Function Api_CreateDirectory& Lib "Kernel32" Alias "CreateDirectoryA" (ByVal
lpPathName$, lpSecurityAttributes As SECURITY_ATTRIBUTES)

' ディレクトリの新規作成
Declare Function Api_CreateDirectoryEx& Lib "Kernel32" Alias "CreateDirectoryExA" (ByVal
lpTemplateDirectory$, ByVal lpNewDirectory$, lpSecurityAttributes As Any)

' ディレクトリを削除
Declare Function Api_RemoveDirectory& Lib "kernel32" Alias "RemoveDirectoryA" (ByVal
lpPathName$)

' ファイルの属性を変更
Declare Function Api_SetFileAttributes& Lib "Kernel32" Alias "SetFileAttributesA" (ByVal
lpFileName$, ByVal dwFileAttributes&)

#define FILE_ATTRIBUTE_ARCHIVE &H20         'アーカイブ属性を示す定数
#define FILE_ATTRIBUTE_HIDDEN &H2           '隠しファイル属性
#define FILE_ATTRIBUTE_NORMAL &H80          '他のファイル属性を持たない
#define FILE_ATTRIBUTE_READONLY &H1         '読み込み専用属性
#define FILE_ATTRIBUTE_SYSTEM &H4           'システムファイル属性

Var Shared Button1 As Object

Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var sa As SECURITY_ATTRIBUTES
    Var Ret As Long

    Ret = Api_CreateDirectory("C:¥TempPath1", sa)
    If Ret <> 0 Then
        A% = MessageBox("", "「C:¥TempPath1」を作成しました！", 0, 2)

        Ret = Api_SetFileAttributes("C:¥TempPath1", FILE_ATTRIBUTE_HIDDEN Or
FILE_ATTRIBUTE_ARCHIVE)

        If Ret <> 0 Then
```

```
            A% = MessageBox("", "「C:¥TempPath1」は、隠しファイル属性に設定しました！", 0, 2)
        End If
    End If

    sa.nLength = Len(sa)
    sa.lpSecurityDescriptor = 0
    sa.bInheritHandle = 1

    Ret = Api_CreateDirectoryEx("C:¥TempPath1", "C:¥TempPath2", sa)

    If Ret <> 0 Then
        A% = MessageBox("", "新ディレクトリ「C:¥TempPath2」を作成しました！", 0, 2)

        If A% = 0 Then
            Ret = Api_RemoveDirectory("C:¥TempPath1")
            Ret = Api_RemoveDirectory("C:¥TempPath2")
            A% = MessageBox("", "「TempPath1・TempPath2」は、削除されました！", 0, 2)
        End If
    End If
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

ディレクトリ・ファイルを列挙

FindFirstFile 指定したファイル名に一致するファイルやディレクトリを検索
FindNextFile FindFirstFile()関数で検出したファイルの次を検出
FindClose ファイル検索ハンドルをクローズ
GetFileAttributes 指定されたファイルまたはディレクトリの属性を取得




```
'=======================================
'= ディレクトリ・ファイルを列挙
'=    (FindFirstFile5.bas)
'=======================================
#include "Windows.bi"

#define MAX_PATH 260
#define API_FALSE 0

#define INVALID_HANDLE_VALUE -1             '見つからない場合
#define ERROR_NO_MORE_FILES 18              'これ以上ファイルは無い
#define vbDirectory 16                      'フォルダ

Type FILETIME
    dwLowDateTime     As Long
    dwHighDateTime    As Long
End Type
```

```
Type WIN32_FIND_DATA
    dwFileAttributes As Long
    ftCreationTime   As FILETIME
    ftLastAccessTime As FILETIME
    ftLastWriteTime  As FILETIME
    nFileSizeHigh    As Long
    nFileSizeLow     As Long
    dwReserved0      As Long
    dwReserved1      As Long
    cFileName        As String * MAX_PATH
    cAlternate       As String * 14
End Type

' 指定したファイル名に一致するファイルやディレクトリを検索
Declare Function Api_FindFirstFile& Lib "Kernel32" Alias "FindFirstFileA" (ByVal
lpFileName$, lpFindFileData As WIN32_FIND_DATA)

' FindFirstFile()関数で検出したファイルの次を検出
Declare Function Api_FindNextFile& Lib "Kernel32" Alias "FindNextFileA" (ByVal
hFindFile&, lpFindFileData As WIN32_FIND_DATA)

' ファイル検索ハンドルをクローズ
Declare Function Api_FindClose& Lib "Kernel32" Alias "FindClose" (ByVal hFindFile&)

' 指定されたファイルまたはディレクトリの属性を取得
Declare Function Api_GetFileAttributes& Lib "Kernel32" Alias "GetFileAttributesA" (ByVal
lpFileName$)

Var Shared Edit1 As Object
Var Shared List1 As Object
Var Shared Button1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
List1.SetWindowSize 252, 132
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Function StripNulls(sText As String) As String
Function StripNulls(sText As String) As String
    Var nPos As Long

    StripNulls = sText
    nPos = InStr(sText, Chr$(0))

    If nPos Then StripNulls = Left$(sText, nPos - 1)
    If Len(sText) Then
        If Left$(sText, 1) = Chr$(0) Then
            StripNulls = ""
        End If
    End If
End Function

'=====================================
'=
'=====================================
Declare Function FileExists(sFileName As String) As Integer
Function FileExists(sFileName As String) As Integer
    Var hFile As Long
    Var wfd As WIN32_FIND_DATA
    Var Ret As Long

    sFileName = Trim$(sFileName)
    hFile = Api_FindFirstFile(sFileName, wfd)

    If (hFile <> INVALID_HANDLE_VALUE) And (hFile <> ERROR_NO_MORE_FILES) Then
        FileExists = True
    Else If Api_GetFileAttributes(sFileName) <> (-1) Then
```

```
        FileExists = True
    End If

    Ret = Api_FindClose(hFile)
End Function

'=====================================
'=
'=====================================
Declare Sub FindFiles(sPath As String)
Sub FindFiles(sPath As String)
    Var wfd As WIN32_FIND_DATA
    Var hFileSearch As Long
    Var sFileName As String
    Var Ret As Long

    If FileExists(sPath) Then
        If Right$(sPath, 1) <> "¥" Then sPath = sPath & "¥"

        hFileSearch = Api_FindFirstFile(sPath & "*.*", wfd)

        If hFileSearch <> INVALID_HANDLE_VALUE Then
            Do
                sFileName = StripNulls(wfd.cFileName)

                If wfd.dwFileAttributes And vbDirectory Then
                    List1.AddString sFileName & " (Dir)"
                Else
                    List1.AddString sFileName
                End If

                If Api_FindNextFile(hFileSearch, wfd) = API_FALSE Then
                    Ret = Api_FindClose(hFileSearch)
                    Exit Do
                End If
            Loop
        Else
            List1.AddString "ファイルは見つかりません！"
        End If
    Else
        List1.AddString "パスは無効です！"
    End If
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Edit1.SetWindowText "C:¥"
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    List1.Resetcontent
    SetMousePointer 2
    FindFiles Edit1.GetWindowText
    SetMousePointer 0
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop : End
```

ディレクトリ名とファイルパスの結合

PathCombine ディレクトリ名とファイルパスの結合


```
'=====================================
'= ディレクトリ名とファイルパスの結合
'=    (PathCombine.bas)
'=====================================
#include "Windows.bi"

' ディレクトリ名とファイルパスの結合
Declare Function Api_PathCombine& Lib "shlwapi" Alias "PathCombineA" (ByVal szDest$,
ByVal lpszDir$, ByVal lpszFile$)

Var Shared Edit1 As Object
Var Shared Edit2 As Object
Var Shared Text1 As Object
Var Shared Button1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Edit2.Attach GetDlgItem("Edit2") : Edit2.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'= Chr$(0)を取り除く
'=====================================
Declare Function TrimNull (item As String) As String
Function TrimNull(item As String) As String
    Var ePos As Integer

    ePos = Instr(item, Chr$(0))
    If ePos Then
        TrimNull = Left$(item, ePos - 1)
    Else
        TrimNull = item
    End If
End Function

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Edit1.SetWindowText "C:"
    Edit2.SetWindowText "One¥Two¥Three"
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Buff As String
    Var path1 As String
    Var path2 As String
    Var Ret As Long

    Buff = String$(100, 0)
    path1 = Edit1.GetWindowText
```

```
    path2 = Edit2.GetWindowText

    Ret = Api_PathCombine(Buff, path1, path2)

    Text1.SetWindowText TrimNull(Buff)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

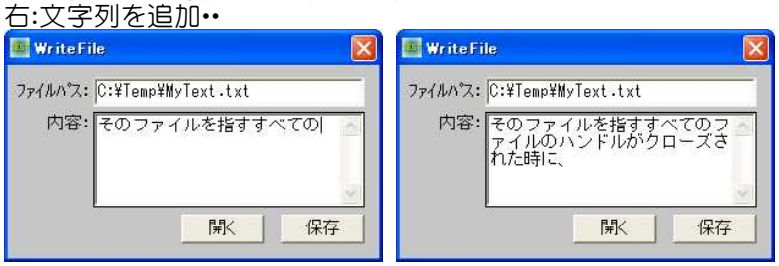
データの読み書き

ファイルからデータを読み込み、またファイルへデータを書き込みます。
CreateFile 指定したファイルをオープンし、デバイスハンドルを返す
ReadFile ファイルからデータを読み取る
WriteFile データをファイルに書き出す
SetFilePointer開いているファイルのポインタを移動
CloseHandle オープンされているオブジェクトハンドルをクローズ

左:初期状態(フォルダC:¥Tempにファイル名MyText.txtを指定)
中:内容を入力
右:「保存」ボタンをクリック

下:確認メッセージ表示(「はい」選択で保存される。一旦「×」で終了させる)

左:再度起動し「開く」をクリック(保存されているのが確認できる)
右:文字列を追加••




```
'=====================================
'= データの読み書き
'=    (WriteFile.bas)
'=====================================
#include "Windows.bi"

' 指定したファイルをオープンし、デバイスハンドルを返す
Declare Function Api_CreateFile& Lib "kernel32" Alias "CreateFileA" (ByVal lpFileName$,
```

```
ByVal dwDesiredAccess&, ByVal dwShareMode&, lpSecurityAttributes As Any, ByVal
dwCreationDisposition&, ByVal dwFlagsAndAttributes&, ByVal hTemplateFile&)

' ファイルからデータを読み取る
Declare Function Api_ReadFile& Lib "kernel32" Alias "ReadFile" (ByVal hFile&, lpBuffer As
Any, ByVal nNumberOfBytesToRead&, lpNumberOfBytesRead&, lpOverlapped&)

' データをファイルに書き出す
Declare Function Api_WriteFile& Lib "kernel32" Alias "WriteFile" (ByVal hFile&, lpBuffer
As Any, ByVal nNumberOfBytesToWrite&, lpNumberOfBytesWritten&, lpOverlapped As Any)

' 開いているファイルのポインタを移動
Declare Function Api_SetFilePointer& Lib "kernel32" Alias "SetFilePointer" (ByVal
hFile&, ByVal lDistanceToMove&, lpDistanceToMoveHigh&, ByVal dwMoveMethod&)

' オープンされているオブジェクトハンドルをクローズ
Declare Function Api_CloseHandle& Lib "kernel32" Alias "CloseHandle" (ByVal hObject&)

#define FILE_BEGIN 0                          'ファイルポインタがファイルの先頭がファイルの先頭からオフセットぶんだけ移動
#define FILE_CURRENT 1                        '現在のファイルポインタの位置
#define FILE_END 2                            'ファイルの終端が移動の開始点
#define FILE_ATTRIBUTE_ARCHIVE &H20           'アーカイブ属性を示す定数
#define FILE_ATTRIBUTE_COMPRETSED &H800       '圧縮属性を示す定数
#define FILE_ATTRIBUTE_DIRECTORY &H10         'ディレクトリ属性
#define FILE_ATTRIBUTE_HIDDEN &H2             '隠しファイル属性
#define FILE_ATTRIBUTE_NORMAL &H80            '他のファイル属性を持たない
#define FILE_ATTRIBUTE_READONLY &H1           '読み込み専用属性
#define FILE_ATTRIBUTE_SYSTEM &H4             'システムファイル属性
#define FILE_ATTRIBUTE_TEMPORARY &H100        '一時ファイル属性を示す定数
#define FILE_CASE_PRESERVED_NAMES &H2         '
#define FILE_CASE_SENSITIVE_SEARCH &H1        '
#define FILE_FILE_COMPRESSION &H10            '
#define FILE_FLAG_DELETE_ON_CLOSE &H4000000       'そのファイルを指すすべてのファイルのハンドルがクローズされた時に、そのファイルを削除するように指定
#define FILE_FLAG_NO_BUFFERING &H20000000 'システムキャッシュを使用せずにファイルをオープンするように指定
#define FILE_FLAG_OVERLAPPED &H40000000   '時間のかかる処理に対してReadFile関数やWriteFile関数でERROR_IO_PENDING拡張エラーを返すようにする
#define FILE_FLAG_POSIX_SEMANTICS &H1000000       'POSIXの規則に従ってファイルにアクセス
#define FILE_FLAG_RANDOM_ACCESS &H10000000        'ファイルにランダムアクセスすることをシステムに示す
#define FILE_FLAG_SEQUENTIAL_SCAN &H8000000       'ファイルにシーケンシャルアクセスすることをシステムに示す
#define FILE_FLAG_WRITE_THROUGH -2147483648       'キャッシュに書き込まれたデータをそのまま直接ディスクに書き込むようにする(&H80000
#define FILE_SHARE_READ &H1                   '後続のオープン操作で読み取りアクセスが要求された場合、そのオープンを許可
#define FILE_SHARE_WRITE &H2                  '後続のオープン操作で書き込みアクセスが要求された場合、そのオープンを許可
#define GENERIC_READ -2147483648              '読み込みモード(&H80000000)
#define GENERIC_WRITE &H40000000              '書き込みモード
#define CREATE_ALWAYS 2                       '新しいファイルを作る(存在する場合には上書)
#define CREATE_NEW 1                          '新しいファイルを作る(存在する場合は失敗)
#define OPEN_ALWAYS 4                         'ファイルをオープンする(存在しない場合作成)
#define OPEN_EXISTING 3                       'ファイルをオープンする(存在しない場合失敗)
#define TRUNCATE_EXISTING 5                   'ファイルをオープンし、ファイルのサイズを0バイトに

Var Shared Edit(1) As Object
Var Shared Text(1) As Object
Var Shared Button(1) As Object
For i = 0 To 1
    Edit(i).Attach GetDlgItem("Edit" & Trim$(Str$(i + 1))) : Edit(i).SetFontSize 14
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1))) : Text(i).SetFontSize 14
    Button(i).Attach GetDlgItem("Button" & Trim$(Str$(i + 1))) : Button(i).SetFontSize 14
Next i
```

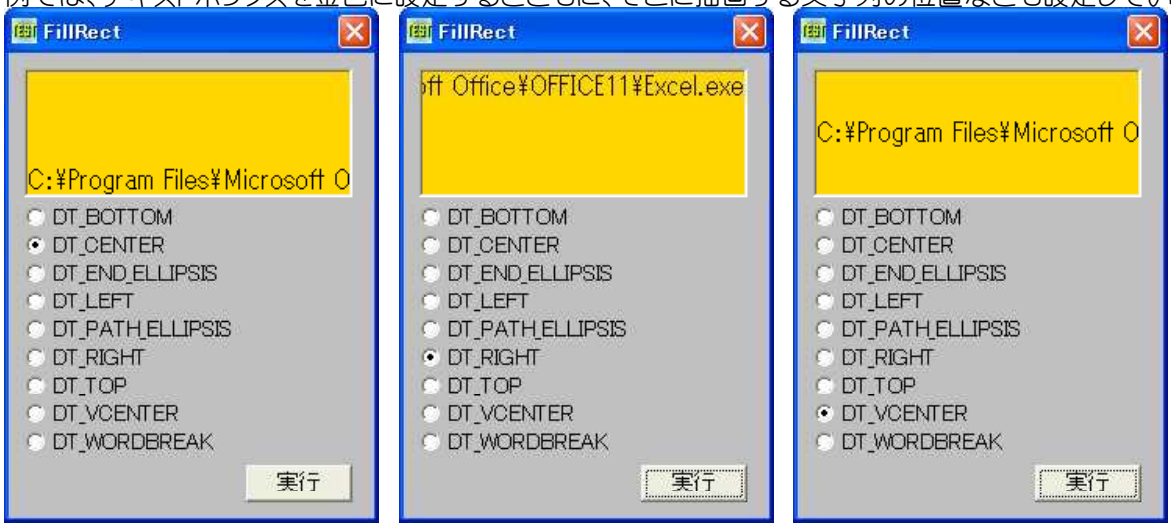
```
'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Edit(0).SetWindowText "C:¥Temp¥MyText.txt"
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hFile As Long
    Var FilePath As String
    Var Buffer As String * 1024
    Var lRead As Long
    Var Ret As Long

    FilePath = Edit(0).GetWindowText
    If FilePath = "" Then Exit Sub

    hFile = Api_CreateFile(FilePath, GENERIC_READ, FILE_SHARE_READ, ByVal 0,
OPEN_ALWAYS, FILE_ATTRIBUTE_ARCHIVE, 0)

    Edit(1).SetWindowText ""
    Do
        Ret = Api_ReadFile(hFile, Buffer, Len(Buffer), lRead, ByVal 0)
        Edit(1).SetWindowText Edit(1).GetWindowText & Left$(Buffer, lRead)
        CallEvent
    Loop until lRead < Len(Buffer)

    Ret = Api_CloseHandle(hFile)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var hFile As Long
    Var FilePath As String
    Var Buffer As String * 1024
    Var lWrite As Long
    Var Ret As Long

    FilePath = Edit(0).GetWindowText
    If FilePath = "" Then Exit Sub

    A% = MessageBox("保存", "保存しますか？", 4, 1)
    If A% = 6 Then Exit Sub
    hFile = Api_CreateFile(FilePath, GENERIC_WRITE, FILE_SHARE_READ, ByVal 0,
TRUNCATE_EXISTING, FILE_ATTRIBUTE_ARCHIVE, 0)

    Ret = Api_SetFilePointer(hFile, 0, ByVal 0, FILE_BEGIN)

    If Len(Edit(1).GetWindowText) > 1024 Then
        For i = 0 To Len(Edit(1).GetWindowText) / 1024
            Buffer = Left$(Edit(1).GetWindowText, 1024)
            Edit(0).SetWindowText Mid$(Edit(1).GetWindowText, 1025)
            Ret = Api_WriteFile(hFile, Buffer, Len(Buffer), lWrite, ByVal 0)
        Next i
    End If
    Ret = Api_WriteFile(hFile, Edit(1).GetWindowText, Len(Edit(1).GetWindowText),
lWrite, ByVal 0)

    Ret = Api_CloseHandle(hFile)
End Sub
```

```
'=====================================
'=
'=====================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
    End
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

テキストボックスの背景色を設定

FillRect ブラシで矩形領域を塗りつぶす
CreateSolidBrush ソリッドカラーで論理ブラシを作成
SetBkMode バックグラウンドの塗りつぶしモード設定
DrawText 文字列を指定領域に出力
GetDC デバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放

例では、テキストボックスを金色に設定するとともに、そこに描画する文字列の位置なども設定しています。



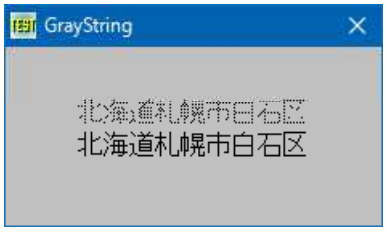

```
'=====================================
'= テキストボックスの背景色を設定
'=    (TextBoxBackColor.bas)
'=====================================
#include "Windows.bi"

Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

' ブラシで矩形領域を塗りつぶす
Declare Function Api_FillRect& Lib "user32" Alias "FillRect" (ByVal hDC&, ByRef r As RECT,
ByVal hBrush&)

' ソリッドカラーで論理ブラシを作成
Declare Function Api_CreateSolidBrush& Lib "gdi32" Alias "CreateSolidBrush" (ByVal
crColor&)
```

```
' バックグラウンドの塗りつぶしモード設定
Declare Function Api_SetBkMode& Lib "gdi32" Alias "SetBkMode" (ByVal hDC&, ByVal
iBkMode&)

' 文字列を指定領域に出力
Declare Function Api_DrawText& Lib "user32" Alias "DrawTextA" (ByVal hDC&, ByVal lpStr$,
ByVal nCount&, lpRect As RECT, ByVal wFormat&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

#define TRANSPARENT 1                    '背景色を設定しない
#define OPAQUE 2                         '背景色を設定する
#define DT_BOTTOM &H8                    '長方形の下辺にテキストを揃える。DT_SINGLELINEと同時
                                         に指定
#define DT_CENTER &H1                    'テキストを水平方向に中央揃えで表示します。
#define DT_END_ELLIPSIS &H8000           '指定した長方形領域にテキストが収まらない場合、テキスト
                                         の最後を (…) に置き換える
#define DT_LEFT &H0                      'テキストを左揃え
#define DT_PATH_ELLIPSIS &H4000          '指定した長方形領域にテキストが収まらない場合、テキスト
                                         の途中を (…) に置き換える
#define DT_RIGHT &H2                     'テキストを右揃え
#define DT_TOP &H0                       '上揃え。DT_SINGLELINEと同時に指定する必要がある
#define DT_SINGLELINE &H20               'テキストを改行せず、一行で表示
#define DT_VCENTER &H4                   'テキストを垂直方向の中央揃え。DT_SINGLELINEと同時に
                                         指定する必要がある
#define DT_WORDBREAK &H10                'テキストを複数行で表示。折り返しは自動的に行われる

Var Shared Text1 As Object
Var Shared Radio(8) As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
For i = 0 To 8
    Radio(i).Attach GetDlgItem("Radio" & Trim$(Str$(i + 1))) : Radio(i).SetFontSize 14
Next
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Function Index bdecl () As Integer
Function Index()
    Index = Val(Mid$(GetDlgRadioSelect("Radio1"), 6)) - 1
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hDC As Long
    Var colBrush As Long
    Var rct As RECT
    Var txt As String
    Var wFormat As Long
    Var Ret As Long

    txt = "C:¥Program Files¥Microsoft Office¥OFFICE11¥Excel.exe"

    hDC = Api_GetDC(Text1.GethWnd)
    colBrush = Api_CreateSolidBrush(RGB(255, 215, 0))

    rct.Top = 0
    rct.Left = 0
    rct.Right = Text1.GetWidth - 4
    rct.Bottom = Text1.GetHeight - 4
```

```
    Ret = Api_FillRect(hDC, rct, colBrush)
    Ret = Api_SetBkMode(hDC, TRANSPARENT)

    Select Case Index
        Case 0
            wFormat = DT_SINGLELINE Or DT_BOTTOM
        Case 1
            wFormat = DT_CENTER
        Case 2
            wFormat = DT_END_ELLIPSIS
        Case 3
            wFormat = DT_LEFT
        Case 4
            wFormat = DT_PATH_ELLIPSIS
        Case 5
            wFormat = DT_RIGHT
        Case 6
            wFormat = DT_SINGLELINE Or DT_TOP
        Case 7
            wFormat = DT_SINGLELINE Or DT_VCENTER
        Case 8
            wFormat = DT_WORDBREAK
    End Select

    Ret = Api_DrawText(hDC, txt, Len(txt), rct, wFormat)

    Ret = Api_ReleaseDC(Text1.GethWnd, hDC)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

テキストをグレー表示で出力

テキストをグレー表示で出力します。その下に同じ文字列を描画しています。
GrayString テキストをグレー表示で出力
SetBkMode バックグラウンドの塗りつぶしモード設定
TextOut 文字を描画
GetDC デバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放



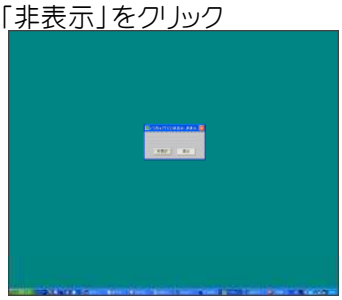

```
'=====================================
'= テキストをグレー表示で出力
'=    (GrayString.bas)
'=====================================
#include "Windows.bi"

' テキストをグレー表示で出力
Declare Function Api_GrayString& Lib "user32" Alias "GrayStringA" (ByVal hDC&, ByVal
hBrush&, ByVal lpOutputFunc&, ByVal lpData$, ByVal nCount&, ByVal X&, ByVal Y&, ByVal
nWidth&, ByVal nHeight&)
```

```
' バックグラウンドの塗りつぶしモード設定
Declare Function Api_SetBkMode& Lib "gdi32" Alias "SetBkMode" (ByVal hDC&, ByVal
iBkMode&)

' 文字を描画
Declare Function Api_TextOut& Lib "gdi32" Alias "TextOutA" (ByVal hDC&, ByVal nXStart&,
ByVal nYStart&, ByVal lpString$, ByVal cbString&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

#define TRANSPARENT 1                          '背景色を設定しない

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var Str As String
    Var hDC As Long
    Var Ret As Long

    Str = "北海道札幌市白石区"
    hDC = Api_GetDC(GethWnd)

    Ret = Api_GrayString(hDC, ByVal 0, ByVal 0, Str, Len(Str), 45, 30, 0, 0)
    Ret = Api_SetBkMode(hDC, TRANSPARENT)
    Ret = Api_TextOut(hDC, 45, 50, Str, Len(Str))
    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

デスクトップアイコンの表示・非表示

デスクトップ上にあるアイコン群の表示・非表示を実行します。
FindWindow クラス名またはキャプションを与えてウィンドウのハンドルを取得
ShowWindow 指定されたウィンドウの表示状態を設定

「非表示」をクリック



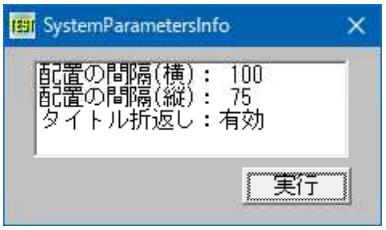
「表示」をクリック

```
'=====================================
'= デスクトップアイコンの表示・非表示
'=    (DesktopIconShowHide.bas)
'=====================================
#include "Windows.bi"
```

```
' クラス名またはキャプションを与えてウィンドウのハンドルを取得
Declare Function Api_FindWindow& Lib "user32" Alias "FindWindowA" (ByVal lpClassName$,
ByVal lpWindowName$)

' 指定されたウィンドウの表示状態を設定
Declare Function Api_ShowWindow& Lib "user32" Alias "ShowWindow" (ByVal hWnd&, ByVal
nCmdShow&)

#define SW_HIDE 0                          '指定のウィンドウを非表示にし他のウィンドウをアクティブ化
#define SW_RESTORE 9                       'ウィンドウをアクティブ化し表示。ウィンドウがアイコン化ま
                                           たは最大化されているときは元の位置とサイズに
Var Shared Button1 As Object
Var Shared Button2 As Object

Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14

'=====================================
'= アイコンを非表示
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hWnd As Long
    Var Ret As Long

    hWnd = Api_FindWindow(ByVal 0, "Program Manager")
    If Not hWnd = 0 Then
        Ret = Api_ShowWindow(hWnd, SW_HIDE)
    End If
End Sub

'=====================================
'= アイコンを表示
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var hWnd As Long
    Var Ret As Long

    hWnd = Api_FindWindow(ByVal 0, "Program Manager")
    If Not hWnd = 0 Then
        Ret = Api_ShowWindow(hWnd, SW_RESTORE)
    End If
    SetFocus                               'フォームをアクティブに
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop : End
```

デスクトップアイコンの表示要素を取得

デスクトップアイコンの間隔およびタイトルの折り返し状態を取得します。
SystemParametersInfo システム全体に関するパラメータを取得・設定



```
'====================================
'= デスクトップアイコンの表示要素を取得
'=    (SystemParametersInfo2.bas)
'====================================
#include "Windows.bi"

#define LF_FACESIZE 32

Type LOGFONT
    lfHeight          As Long
    lfWidth           As Long
    lfEscapement      As Long
    lfOrientation     As Long
    lfWeight          As Long
    lfItalic          As Byte
    lfUnderline       As Byte
    lfStrikeOut       As Byte
    lfCharSet         As Byte
    lfOutPrecision    As Byte
    lfClipPrecision   As Byte
    lfQuality         As Byte
    lfPitchAndFamily As Byte
    lfFaceName(LF_FACESIZE - 1) As Byte
End Type

Type ICONMETRICS
    cbSize            As Long
    iHorzSpacing      As Long
    iVertSpacing      As Long
    iTitleWrap        As Long
    lfFont            As LOGFONT
End Type

#define SPI_GETICONMETRICS 45                'アイコンに関する寸法情報を定義するICONMETRICS構造体取得
' システム全体に関するパラメータを取得・設定
Declare Function Api_SystemParametersInfo& Lib "user32" Alias "SystemParametersInfoA"
(ByVal uiAction&, ByVal uiParam&, pvParam As Any, ByVal fWinIni&)

Var Shared List1 As Object
Var Shared Button1 As Object

List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'====================================
'=
'====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var im As ICONMETRICS
    Var iFontName As String
    Var Ret As Long

    'リストボックスをクリア
    List1.Resetcontent

    '構造体を初期化
    im.cbSize = Len(im)

    'アイコンの表示要素を取得
    Ret = Api_SystemParametersInfo(SPI_GETICONMETRICS, Len(im), im, 0)

    'アイコンの表示要素を表示
    List1.AddString "配置の間隔(横):" & Str$(im.iHorzSpacing)
    List1.AddString "配置の間隔(縦):" & Str$(im.iVertSpacing)

    If im.iTitleWrap <> False Then
        List1.AddString "タイトル折返し:" & "有効"
    Else
```

```
        List1.AddString "タイトル折返し:" & "無効"
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

デスクトップイメージを異なる方法で転送

デスクトップ画面をBitBlt、StretchBlt、SetStretchBltModeで転送しています。
BitBlt ビットブロック転送を行う
StretchBlt 拡縮をともなうグラフィックデバイス間のイメージを転送
SetStretchBltMode 指定されたデバイスコンテキストのビットマップ伸縮モードを設定
GetDesktopWindow Windows のデスクトップ ウィンドウを識別
GetWindowDC ウィンドウ全体のデバイスコンテキストを取得
GetDC デバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放

左:BitBlt(等倍・そのまま転送) 中:StretchBltMode(綺麗に縮小転送) 右:StretchBlt(通常縮小転送)


```
'=====================================
'= デスクトップ画面を転送
'=    (BitBlt2.bas)
'=====================================
#include "Windows.bi"

' ビットブロック転送を行う。コピー元からコピー先のデバイスコンテキストへ、指定された長方形内の各ピクセルの色データをコピー
Declare Function Api_BitBlt& Lib "gdi32" Alias "BitBlt" (ByVal hDestDC&, ByVal X&, ByVal Y&, ByVal nWidth&, ByVal nHeight&, ByVal hSrcDC&, ByVal xSrc&, ByVal ySrc&, ByVal dwRop&)

' 拡縮をともなうグラフィックデバイス間のイメージを転送
Declare Function Api_StretchBlt& Lib "gdi32" Alias "StretchBlt" (ByVal hDC&, ByVal X&, ByVal Y&, ByVal nWidth&, ByVal nHeight&, ByVal hSrcDC&, ByVal xSrc&, ByVal ySrc&, ByVal nSrcWidth&, ByVal nSrcHeight&, ByVal dwRop&)

' 指定されたデバイスコンテキストのビットマップ伸縮モードを設定
Declare Function Api_SetStretchBltMode& Lib "gdi32" Alias "SetStretchBltMode" (ByVal hDC&, ByVal nStretchMode&)

' Windows のデスクトップ ウィンドウを識別。返されるポインタは、一時的なポインタ。後で使用するために保存しておくことはできない
Declare Function Api_GetDesktopWindow& Lib "user32" Alias "GetDesktopWindow" ()

' ウィンドウ全体のデバイスコンテキストを取得
Declare Function Api_GetWindowDC& Lib "user32" Alias "GetWindowDC" (ByVal hWnd&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)
```

```
' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

#define SRCCOPY &HCC0020                    '単純な上書き
#define COLORONCOLOR 3                      '取り除く点の情報を保存することなく削除

Var Shared PICTURE1 As Object
Var Shared Button1 As Object
Var Shared Button2 As Object
Var Shared Button3 As Object

Picture1.Attach GetDlgItem("Picture1")
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14
Button3.Attach GetDlgItem("Button3") : Button3.SetFontSize 14
Var Shared dWidth As Long
Var Shared dHeight As Long
Var Shared Flg As byte

'=====================================
'=
'=====================================
Declare Sub Mainform_Start edecl ()
Sub Mainform_Start()
    dWidth = GetDeviceCaps(8)
    dHeight = GetDeviceCaps(10)
End Sub

'=====================================
'=
'=====================================
Declare Sub PrintScreen edecl ()
Sub PrintScreen()
    Var dHwnd As Long
    Var dhDC As Long
    Var phDC As Long
    Var dLeft As Long
    Var dTop As Long
    Var Ret As Long
    dLeft = 0
    dTop = 0

    dWidth = GetDeviceCaps(8)
    dHeight = GetDeviceCaps(10)

    dHwnd = Api_GetDesktopWindow()
    dhDC = Api_GetWindowDC(dHwnd)
    phDC = Api_GetDC(Picture1.GethWnd)


    If Flg = 0 Then
        Ret = Api_BitBlt(phDC, 0, 0, dWidth, dHeight, dhDC, dLeft, dTop, SRCCOPY)
    Else If Flg = 1 Then
        Ret = Api_SetStretchBltMode(phDC, COLORONCOLOR)
        Ret = Api_StretchBlt(phDC, 0, 0, Picture1.GetWidth, Picture1.GetHeight, dhDC,
dLeft, dTop, dWidth, dHeight, SRCCOPY)
    Else
        Ret = Api_StretchBlt(phDC, 0, 0, Picture1.GetWidth, Picture1.GetHeight, dhDC,
dLeft, dTop, dWidth, dHeight, SRCCOPY)
    End If

    Ret = Api_ReleaseDC(dHwnd, dhDC)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Flg = 0
```

```
    PrintScreen
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Flg = 1
    PrintScreen
End Sub

'=====================================
'=
'=====================================
Declare Sub Button3_on edecl ()
Sub Button3_on()
    Flg = 2
    PrintScreen
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

デスクトップイメージを転送

デスクトップイメージを転送します。
CreateCompatibleDC 指定されたデバイスコンテキストに関連するデバイスと互換性のあるメモリデバイスコンテキストを作成
CreateDIBSection プリケーションから直接書き込むことのできるDIBを作成
GetDIBits 指定されたビットマップのビットを取得し、指定された形式でバッファへコピー
SetDIBitsToDevice DIBの色データを使って、指定された長方形内のピクセルを、転送先のデバイスコンテキストに関連付けられているデバイスの指定された長方形内に描画
SelectObject 指定されたデバイスコンテキストのオブジェクトを選択
DeleteDC 指定されたデバイスコンテキストを削除
DeleteObject オブジェクトに関連付けられていたすべてのシステムリソースを解放
BitBlt ビットブロック転送を行う。コピー元からコピー先のデバイスコンテキストへ、指定された長方形内の各ピクセルの色データをコピー
GetDC 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放


```
'=====================================
'= デスクトップイメージを転送
'=    (SetDIBitsToDevice.bas)
'=====================================
#include "Windows.bi"

#define BI_RGB 0                              '非圧縮
```

```
#define DIB_RGB_COLORS 0                    'RGBカラーテーブル
#define SRCCOPY &HCC0020                    'そのまま転送

Type BITMAPINFOHEADER                       '40バイト
    biSize            As Long               'イメージ バッファのバイト数
    biWidth           As Long               '幅
    biHeight          As Long               '高さ
    biPlanes          As Integer            '常に1
    biBitCount        As Integer            '1ピクセルあたりのカラービット数
    biCompression     As Long               '圧縮方法
    biSizeImage       As Long               'ピクセルデータの全バイト数
    biXPelsPerMeter   As Long               '0または水平解像度
    biYPelsPerMeter   As Long               '0または垂直解像度
    biClrUsed         As Long               'ビットマップを表示するための色数(0)
    biClrImportant    As Long               'ビットマップを表示するための重要な色数(0)
End Type

Type RGBQUAD
    rgbBlue           As Byte               '青の輝度
    rgbGreen          As Byte               '緑の輝度
    rgbRed            As Byte               '赤の輝度
    rgbReserved       As Byte               '予約(常に0)
End Type

Type BITMAPINFO
    bmiHeader As BITMAPINFOHEADER
    bmiColors As RGBQUAD                    '256色の場合、bmiColors(255) As RGBQUADとする
End Type

' 指定されたデバイスコンテキストに関連するデバイスと互換性のあるメモリデバイスコンテキストを作成
Declare Function Api_CreateCompatibleDC& Lib "gdi32" Alias "CreateCompatibleDC" (ByVal
hDC&)

' アプリケーションから直接書き込むことのできるDIBを作成
Declare Function Api_CreateDIBSection& Lib "gdi32" Alias "CreateDIBSection" (ByVal hDC&,
pBitmapInfo As BITMAPINFO, ByVal un&, ByVal lplpVoid&, ByVal handle&, ByVal dw&)

' DIBのピクセルデータを取得
Declare Function Api_GetDIBits& Lib "gdi32" Alias "GetDIBits" (ByVal aHDC&, ByVal
hBitmap&, ByVal nStartScan&, ByVal nNumScans&, lpBits As Any, lpBI As BITMAPINFO, ByVal
wUsage&)

' DIBをデバイスコンテキスト上の矩形領域に描画
Declare Function Api_SetDIBitsToDevice& Lib "gdi32" Alias "SetDIBitsToDevice" (ByVal
hDC&, ByVal x&, ByVal y&, ByVal dx&, ByVal dy&, ByVal SrcX&, ByVal SrcY&, ByVal Scan&,
ByVal NumScans&, Bits As Any, BitsInfo As BITMAPINFO, ByVal wUsage&)

' 指定されたデバイスコンテキストのオブジェクトを選択
Declare Function Api_SelectObject& Lib "gdi32" Alias "SelectObject" (ByVal hDC&, ByVal
hObject&)

' 指定されたデバイスコンテキストを削除
Declare Function Api_DeleteDC& Lib "gdi32" Alias "DeleteDC" (ByVal hDC&)

' ペン、ブラシ、フォント、ビットマップ、リージョン、パレットのいずれかの論理オブジェクトを削除し、そのオブジェクトに
関連付けられていたすべてのシステムリソースを解放。オブジェクトを削除した後は、指定されたハンドルは無効になる
Declare Function Api_DeleteObject& Lib "gdi32" Alias "DeleteObject" (ByVal hObject&)

' ビットブロック転送を行う。コピー元からコピー先のデバイスコンテキストへ、指定された長方形内の各ピクセルの色
データをコピー
Declare Function Api_BitBlt& Lib "gdi32" Alias "BitBlt" (ByVal hDestDC&, ByVal X&, ByVal
Y&, ByVal nWidth&, ByVal nHeight&, ByVal hSrcDC&, ByVal xSrc&, ByVal ySrc&, ByVal dwRop&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

Var Shared Button1 As Object
```

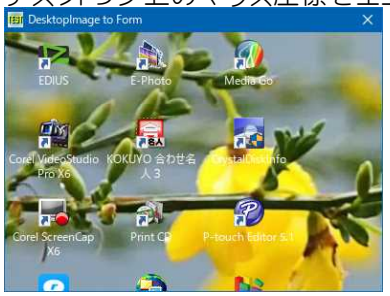
```
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var Shared iBitmap As Long
Var Shared iDC As Long

ShowWindow -1
Cls

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hDC As Long
    Var bi As BITMAPINFO
    Var Cnt As Long
    Var Ret As Long

    hDC = Api_GetDC(GethWnd)

    bi.bmiHeader.biSize = Len(bi.bmiHeader)
    bi.bmiHeader.biWidth = 300
    bi.bmiHeader.biHeight = 250
    bi.bmiHeader.biPlanes = 1
    bi.bmiHeader.biBitCount = 24
    bi.bmiHeader.biCompression = BI_RGB

    '幅 * 高さ * 3(RGB)
    Var bBytes(bi.bmiHeader.biWidth * bi.bmiHeader.biHeight * 3) As Byte

    '「0」の場合、ディスプレイのデバイスコンテキスト
    iDC = Api_CreateCompatibleDC(0)
    iBitmap = Api_CreateDIBSection(iDC, bi, DIB_RGB_COLORS, ByVal 0, ByVal 0, ByVal 0)

    Ret = Api_SelectObject(iDC, iBitmap)
    Ret = Api_BitBlt(iDC, 0, 0, bi.bmiHeader.biWidth, bi.bmiHeader.biHeight,
Api_GetDC(0), 0, 0, SRCCOPY)
    Ret = Api_GetDIBits(iDC, iBitmap, 0, bi.bmiHeader.biHeight, bBytes(1), bi,
DIB_RGB_COLORS)

    For Cnt = 1 To (bi.bmiHeader.biWidth * bi.bmiHeader.biHeight * 3)
        If bBytes(Cnt) < 50 Then
            bBytes(Cnt) = 0
        Else
            bBytes(Cnt) = bBytes(Cnt) - 50
        End If
    Next Cnt

    Ret = Api_SetDIBitsToDevice(hDC, 0, 0, bi.bmiHeader.biWidth, bi.bmiHeader.biHeight,
0, 0, 0, bi.bmiHeader.biHeight, bBytes(1), bi, DIB_RGB_COLORS)
    Ret = Api_DeleteDC(iDC)
    Ret = Api_DeleteObject(iBitmap)
    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

デスクトップイメージをフォームに転送

デスクトップイメージをフォームに転送します。

BitBlt ビットブロック転送
GetDesktopWindow ウィンドウハンドルを取得する関数
GetCursorPos マウスカーソル位置を取得する
GetDC ウィンドウのデバイスコンテキストのハンドルを取得する関数
ReleaseDC デバイスコンテキストの開放

デスクトップ上のマウス座標を左上としたイメージをフォームに転送します。

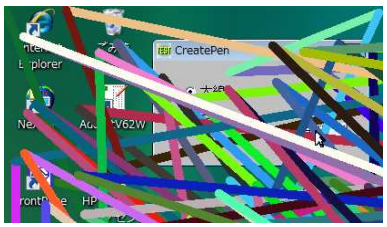


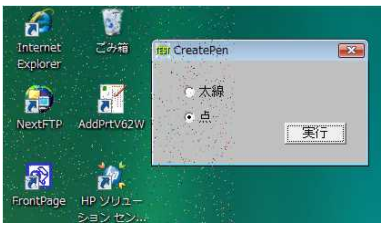

```
'=======================================
'= デスクトップの画像をフォームに転送
'=    (DesktopImageToForm.bas)
'=======================================
#include "Windows.bi"

'点を示すタイプ
Type POINTAPI
    X   As Long
    Y   As Long
End Type

' ビットブロック転送
Declare Function Api_BitBlt& Lib "gdi32" Alias "BitBlt" (ByVal hDestDC&, ByVal X&, ByVal
Y&, ByVal nWidth&, ByVal nHeight&, ByVal hSrcDC&, ByVal xSrc&, ByVal ySrc&, ByVal dwRop&)

' ウィンドウハンドルを取得する関数
Declare Function Api_GetDesktopWindow& Lib "user32" Alias "GetDesktopWindow" ()

'マウスカーソル位置を取得する
Declare Function Api_GetCursorPos& Lib "user32" Alias "GetCursorPos" (lpPoint As
POINTAPI)

' ウィンドウのデバイスコンテキストのハンドルを取得する関数
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストの開放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

#define SRCCOPY &HCC0020                    'コピー元をそのままコピー

Var Shared Timer1 As Object
Timer1.Attach GetDlgItem("Timer1")

'=======================================
'=
'=======================================
Declare Sub Mainform_Start edecl ()
Sub Mainform_Start()
    Timer1.SetInterval 10
    Timer1.Enable -1
End Sub

'=======================================
'= 画面のイメージをフォームにコピー
'=======================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var mXY As POINTAPI
    Var dTophWnd As Long
    Var dTopDC As Long
    Var FormDC As Long
```

```
    Var Ret As Long

    dTophWnd = Api_GetDesktopWindow()       'デスクトップハンドル取得
    dTopDC = Api_GetDC(dTophWnd)            'デスクトップデバイスコンテキスト取得
    FormDC = Api_GetDC(GethWnd)             'フォームデバイスコンテキスト取得
    Ret = Api_GetCursorPos(mXY)
    If mXY.X > GetDeviceCaps(8) - GetClientWidth Then mXY.X = GetDeviceCaps(8) -
GetClientWidth
    If mXY.Y > GetDeviceCaps(10) - GetClientHeight Then mXY.Y = GetDeviceCaps(10) -
GetClientHeight

    Ret = Api_BitBlt(FormDC, 0, 0, GetClientWidth, GetClientHeight, dTopDC, mXY.X, mXY.Y,
SRCCOPY)
    Ret = Api_ReleaseDC(dTophWnd, dTopDC)
    Ret = Api_ReleaseDC(GethWnd, FormDC)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

デスクトップに線・点を描画

CreatePen 論理ペンを作成
LineTo 現在の位置から終点までを直線で描画
CreateDC 指定されたデバイスのデバイスコンテキストを、指定された名前で作成
CreateCompatibleDC デバイスと互換性のあるメモリデバイスコンテキストを作成
DeleteDC 指定されたデバイスコンテキストを削除
SelectObject 指定されたデバイスコンテキストのオブジェクトを選択
DeleteObject 論理オブジェクトを削除し、そのオブジェクトに関連付けられていた全てのシステムリソースを解放
SetPixel 指定した座標に点を配置
CreateCompatibleBitmap デバイスコンテキストと互換性のあるビットマップを作成
BitBlt コピー元からコピー先のデバイスコンテキストへ、指定された長方形内の各ピクセルの色データをコピー






```
'=====================================
'= デスクトップに線・点を描画
'=    (CreatePen3.bas)
'=====================================
#include "Windows.bi"

' 論理ペンを作成
Declare Function Api_CreatePen& Lib "gdi32" Alias "CreatePen" (ByVal nPenStyle&, ByVal
nWidth&, ByVal crColor&)

' 現在の位置から終点までを直線で描画
Declare Function Api_LineTo& Lib "gdi32" Alias "LineTo" (ByVal hDC&, ByVal x&, ByVal y&)

' 指定されたデバイスのデバイスコンテキストを、指定された名前で作成
Declare Function Api_CreateDC& Lib "gdi32" Alias "CreateDCA" (ByVal lpDriverName$, ByVal
lpDeviceName$, ByVal lpOutput$, ByVal lpInitData As Any)

' 指定されたデバイスコンテキストに関連するデバイスと互換性のあるメモリデバイスコンテキストを作成
Declare Function Api_CreateCompatibleDC& Lib "gdi32" Alias "CreateCompatibleDC" (ByVal
```

```
hDC&)

' 指定されたデバイスコンテキストを削除
Declare Function Api_DeleteDC& Lib "gdi32" Alias "DeleteDC" (ByVal hDC&)

' 指定されたデバイスコンテキストのオブジェクトを選択
Declare Function Api_SelectObject& Lib "gdi32" Alias "SelectObject" (ByVal hDC&, ByVal
hObject&)

' 論理オブジェクトを削除し、そのオブジェクトに関連付けられていたすべてのシステムリソースを解放
Declare Function Api_DeleteObject& Lib "gdi32" Alias "DeleteObject" (ByVal hObject&)

' 指定した座標に点を配置する
Declare Function Api_SetPixel& Lib "gdi32" Alias "SetPixel" (ByVal hDC&, ByVal X&, ByVal
Y&, ByVal crColor&)

' デバイスコンテキストと互換性のあるビットマップを作成
Declare Function Api_CreateCompatibleBitmap& Lib "gdi32" Alias "CreateCompatibleBitmap"
(ByVal hDC&, ByVal nWidth&, ByVal nHeight&)

' ビットブロック転送を行う。コピー元からコピー先のデバイスコンテキストへ、指定された長方形内の各ピクセルの色
データをコピー
Declare Function Api_BitBlt& Lib "gdi32" Alias "BitBlt" (ByVal hDestDC&, ByVal X&, ByVal
Y&, ByVal nWidth&, ByVal nHeight&, ByVal hSrcDC&, ByVal xSrc&, ByVal ySrc&, ByVal dwRop&)

#define SRCCOPY &HCC0020                    'そのまま転送

Var Shared Radio1 As Object
Var Shared Radio2 As Object
Var Shared Button1 As Object

Radio1.Attach GetDlgItem("Radio1") : Radio1.SetFontSize 14
Radio2.Attach GetDlgItem("Radio2") : Radio2.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Function Index bdecl () As Integer
Function Index()
    Index = Val(Mid$(GetDlgRadioSelect("Radio1"), 6)) -1
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hDC As Long
    Var hDCBuffer As Long
    Var hBmp As Long
    Var hPen As Long
    Var hObject As Long
    Var DeskWidth As Long
    Var DeskHeight As Long
    Var i As Long
    Var Col As Long
    Var Max As Long
    Var Ret As Long

    Max = 5000 + Index * 100000

    'デスクトップサイズを取得
    DeskWidth = GetDeviceCaps(8)
    DeskHeight = GetDeviceCaps(10)

    'デスクトップにビットマップを描画
    hDC = Api_CreateDC("DISPLAY", ByVal 0, ByVal 0, 0)
    hDCBuffer = Api_CreateCompatibleDC(hDC)
    hBmp = Api_CreateCompatibleBitmap(hDC, DeskWidth, DeskHeight)
```

```
    Ret = Api_SelectObject(hDCBuffer, hBmp)
    Ret = Api_BitBlt(hDCBuffer, 0, 0, DeskWidth, DeskHeight, hDC, 0, 0, SRCCOPY)

    '色・線・点を設定
    For i = 1 To Max
        Col = RGB(CInt(255 * Rnd), CInt(255 * Rnd), CInt(255 * Rnd))
        If Index = 0 Then
            hPen = Api_CreatePen(0, 10, Col)
            hObject = Api_SelectObject(hDC, hPen)
            Ret = Api_LineTo(hDC, CInt(DeskWidth * Rnd), CInt(DeskHeight * Rnd))
            Ret = Api_DeleteObject(hObject)
        Else
            Ret = Api_SetPixel(hDC, CInt(DeskWidth * Rnd), CInt(DeskHeight * Rnd), Col)
        End If
    Next i

    'デスクトップに描画
    Ret = Api_BitBlt(hDC, 0, 0, DeskWidth, DeskHeight, hDCBuffer, 0, 0, SRCCOPY)

    'デバイスコンテキストの解放
    Ret = Api_DeleteDC(hDCBuffer)
    Ret = Api_DeleteDC(hDC)
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

デスクトップに文字列を描画

デスクトップに文字列を描画します。
CreateDC 指定されたデバイスのデバイスコンテキストを、指定された名前で作成
DeleteDC 指定されたデバイスコンテキストを削除
SetBkMode バックグラウンドの塗りつぶしモード設定
DrawText 文字列を指定領域に出力
SetTextColor デバイスコンテキストの文字色を変更

文字色を指定し実行ボタンをクリックすると、指定スクリーン矩形領域に文字列を描画します。


```
'=======================================
'= デスクトップに文字列を描画
'=     (DrawText.bas)
'=======================================
#include "Windows.bi"

Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

#define TRANSPARENT 1                       '背景色を設定しない
```

```
' 指定されたデバイスのデバイスコンテキストを、指定された名前で作成
Declare Function Api_CreateDC& Lib "gdi32" Alias "CreateDCA" (ByVal lpDriverName$,
lpDeviceName As Any, lpOutput As Any, ByVal lpInitData As Any)

' 指定されたデバイスコンテキストを削除
Declare Function Api_DeleteDC& Lib "gdi32" Alias "DeleteDC" (ByVal hDC&)

' バックグラウンドの塗りつぶしモード設定
Declare Function Api_SetBkMode& Lib "gdi32" Alias "SetBkMode" (ByVal hDC&, ByVal
iBkMode&)

' 文字列を指定領域に出力
Declare Function Api_DrawText& Lib "user32" Alias "DrawTextA" (ByVal hDC&, ByVal lpStr$,
ByVal nCount&, lpRect As RECT, ByVal wFormat&)

' デバイスコンテキストの文字色を変更
Declare Function Api_SetTextColor& Lib "gdi32" Alias "SetTextColor" (ByVal hDC&, ByVal
crColor&)

Var Shared Group1 As Object
Var Shared Radio(2) As Object
Var Shared Button1 As Object

'=======================================
'=
'=======================================
Declare Function Index bdecl () As Integer
function Index()
    Index = Val(Mid$(GetDlgRadioSelect("Radio1"), 6)) - 1
End Function

'=======================================
'=
'=======================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Group1.Attach GetDlgItem("Group1") : Group1.SetFontSize 14
    For i = 0 To 2
        Radio(i).Attach GetDlgItem("Radio" & Trim$(Str$(i + 1))) : Radio(i).SetFontSize
14
    Next
    Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

    ShowWindow -1
End Sub

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hDC As Long
    Var rct As RECT
    Var Col As Long
    Var Str As String
    Var Ret As Long

    Str = "F-Basic Programming Tips"

    'デスクトップのDC取得(第一引数を"Display"とした場合他の引数は"Null")
    hDC = Api_CreateDC("DISPLAY", ByVal 0, ByVal 0, ByVal 0)

    '文字を描画する矩形領域
    rct.Left = 30
    rct.Top = 54
    rct.Right = 400
    rct.Bottom = 100

    Select Case Index
        Case 0
```

```
            Col = &HFF                          'Red
        Case 1
            Col = &HFF00                        'Green
        Case 2
            Col = &HFF0000                      'Blue
    End Select

    Ret = Api_SetBkMode(hDC, TRANSPARENT)
    Ret = Api_SetTextColor(hDC, Col)
    Ret = Api_DrawText(hDC, Str, Len(Str), rct, 0)

    Ret = Api_DeleteDC(hDC)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

デスクトップの模様を設定

WriteProfileString WIN.INIの指定のセクションの内容を変更
SystemParametersInfo システム全体に関するパラメータを取得・設定

デスクトップのパターンを10回ランダムに設定し、最後に「子犬」風パターンを設定しています。
文字列 = "64 192 200 120 120 72 0 0" とは
2進数で表すと図のパターンになります。

10進数	2進数	ビットパターン
64	1000000	■
192	11000000	■■
200	11001000	■■ ■
120	1111000	■■■■
120	1111000	■■■■
72	1001000	■ ■
0	0	
0	0	


```
'=====================================
'= デスクトップの模様を設定(Windows10では変化無し)
'=     (SETDESKPATTERN.bas)
'=====================================
#include "Windows.bi"

' WIN.INIの指定のセクションの内容を変更
Declare Function Api_WriteProfileString& Lib "Kernel32" Alias "WriteProfileStringA"
(ByVal lpszSection$, ByVal lpszKeyName$, ByVal lpszString$)

' システム全体に関するパラメータを取得・設定
Declare Function Api_SystemParametersInfo& Lib "user32" Alias "SystemParametersInfoA"
(ByVal uiAction&, ByVal uiParam&, pvParam As Any, ByVal fWinIni&)
```

```
#define SPI_SETDESKPATTERN 21                '
#define SPIF_SENDWININICHANGE &H2            '全てのアプリケーションに通知して更新
#define SPIF_UPDATEINIFILE &H1               'ユーザープロファイルの更新を指定
Var Shared Button1 As Object
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var SectionName As String
    Var KeyName As String
    Var WriteString As String
    Var i As Integer
    Var Ret As Long

    'セクション名を指定
    SectionName = "Desktop"

    'キー名を指定
    KeyName = "Pattern"

    '書き込む文字列を指定
    For i = 1 To 10
        WriteString = ""
        WriteString = WriteString & Trim$(Str$(Rnd(1))) & " "
        WriteString = WriteString & Trim$(Str$(Rnd(1))) & " "
        WriteString = WriteString & Trim$(Str$(Rnd(1))) & " "
        WriteString = WriteString & Trim$(Str$(Rnd(1))) & " "
        WriteString = WriteString & "0 0"
        Gosub *SetPattern
        Wait 50
    Next

    '子犬のパターン文字列
    WriteString = "64 192 200 120 120 72 0 0"
    Gosub *SetPattern
    Exit Sub

*SetPattern

    '指定セクションに文字列を書き込み
    Ret = Api_WriteProfileString(SectionName, KeyName, WriteString)

    'デスクトップの模様を変更
    Ret = Api_SystemParametersInfo(SPI_SETDESKPATTERN, 0, ByVal CLng(0),
SPIF_SENDCHANGE)
    Return
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var SectionName As String
    Var KeyName As String
    Var WriteString As String
    Var Ret As Long

    'セクション名を指定
    SectionName = "Desktop"

    'キー名を指定
    KeyName = "Pattern"
    'パターン文字列(クリア)
    WriteString = ""
```

```
    '指定セクションに文字列を書き込み
    Ret = Api_WriteProfileString(SectionName, KeyName, WriteString)

    'デスクトップの模様を変更
    Ret = Api_SystemParametersInfo(SPI_SETDESKPATTERN, 0, ByVal CLng(0),
SPIF_SENDCHANGE)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

デスクトップのワークエリアを設定

SystemParametersInfo システム全体に関するパラメータを取得・設定

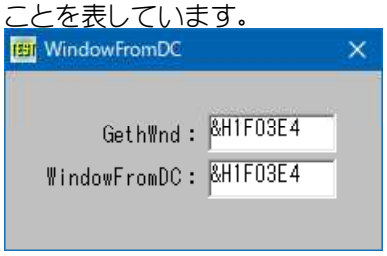

```
'=====================================
'= デスクトップのワークエリアを設定
'=    (SystemParametersInfo4.bas)
'=====================================
#include "Windows.bi"

#define LF_FACESIZE 32

Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

#define SPI_GETWORKAREA 48                  '主モニターの有効なスクリーンのサイズを取得
#define SPI_SETWORKAREA 47                  '主モニターの有効なスクリーンのサイズを設定
#define SPIF_SENDWININICHANGE &H2           '全てのアプリケーションに通知して更新
#define SPIF_UPDATEINIFILE &H1              'ユーザープロファイルの更新を指定

' システム全体に関するパラメータを取得・設定
Declare Function Api_SystemParametersInfo& Lib "user32" Alias "SystemParametersInfoA"
(ByVal uiAction&, ByVal uiParam&, pvParam As Any, ByVal fWinIni&)

Var Shared Text(1) As Object
Var Shared Button(1) As Object

For i = 0 To 1
```

```
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1))) : Text(i).SetFontSize 14
    Button(i).Attach GetDlgItem("Button" & Trim$(Str$(i + 1))) : Button(i).SetFontSize 14
Next i
Var Shared rcOrg As RECT

'=====================================
'=
'=====================================
Declare Sub MainFOrm_Start edecl ()
Sub MainForm_Start()
    Text(1).SetWindowText "(0,0)-(640,480)"

    'ワークエリアを取得
    Ret = Api_SystemParametersInfo(SPI_GETWORKAREA, 0, rcOrg, 0)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var rc As RECT
    Var Ret As Long

    'ワークエリアを指定
    rc.Left = 0
    rc.Top = 0
    rc.Right = 640
    rc.Bottom = 480

    'ワークエリアを表示
    Text(1).SetWindowText "(" & Str$(rc.Left) & "," & Str$(rc.Top) & ")-" & "(" &
Str$(rc.Right) & "," & Str$(rc.Bottom) & ")"

    'ワークエリアを設定
    Ret = Api_SystemParametersInfo(SPI_SETWORKAREA, 0, rc, SPIF_UPDATEINIFILE Or
SPIF_SENDWININICHANGE)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Ret As Long

    'ワークエリアを表示
    Text(1).SetWindowText "(" & Str$(rcOrg.Left) & "," & Str$(rcOrg.Top) & ")-" & "(" &
Str$(rcOrg.Right) & "," & Str$(rcOrg.Bottom) & ")"

    'ワークエリアを設定
    Ret = Api_SystemParametersInfo(SPI_SETWORKAREA, 0, rcOrg, SPIF_UPDATEINIFILE Or
SPIF_SENDWININICHANGE)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

デバイスコンテキストからウィンドウハンドルを取得

WindowFromDC デバイスコンテキストからウィンドウハンドルを取得

例では、GethWnd(フォームのハンドル)とデバイスコンテキストから取得(WindowFromDC)したハンドルが同じであることを表しています。

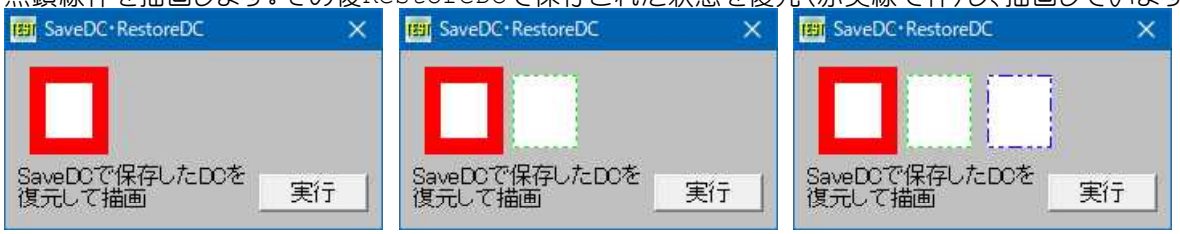



```
'=====================================
'= デバイスコンテキストからウィンドウハンドルを取得
'=    (WindowFromDC.bas)
'=====================================
#include "Windows.bi"

' デバイスコンテキストからウィンドウハンドルを取得
Declare Function Api_WindowFromDC& Lib "user32" Alias "WindowFromDC" (ByVal dDC&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

Var Shared Text(3) As Object
For i = 0 To 3
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1)))
    Text(i).SetFontSize 14
Next

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var hDC As Long
    Var Ret As Long

    'ウィンドウ(メインフォーム)のハンドルを取得表示
    Text(2).SetWindowText "&&H" & Hex$(GethWnd)

    'デバイスコンテキストを取得
    hDC = Api_GetDC(GethWnd)

    'デバイスコンテキストからウィンドウハンドルを取得
    Ret = Api_WindowFromDC(hDC)

    '取得したハンドルを表示(GethWndと同じ)
    Text(3).SetWindowText "&&H" & Hex$(Ret)

    'デバイスコンテキストの解放
    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

デバイスコンテキストの状態保存と復元(Ⅰ)

SaveDC 指定のデバイスコンテキストの現在の状態を保存
RestoreDC SaveDCで保存したデバイスコンテキストを復元
CreatePen 論理ペンを作成
SelectObject 指定されたデバイスコンテキストのオブジェクトを選択
Rectangle 長方形の描画
DeleteObject 論理オブジェクトを削除し、関連システムリソースを解放
GetDC デバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストの解放

例では、最初太い赤実線で枠を描画し、そのデバイスコンテキストの状態を保存しておきます。次に緑の点線枠、青の二点鎖線枠を描画します。その後RestoreDCで保存された状態を復元(赤実線で枠)し、描画しています。

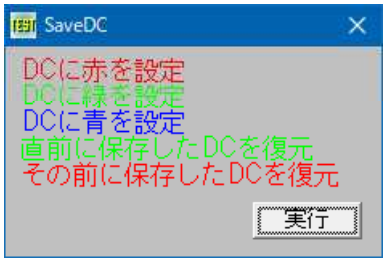



```
'=====================================
'= デバイスコンテキストの状態保存と復元
'=    (RestoreDC2.bas)
'=====================================
#include "Windows.bi"

' 指定のデバイスコンテキストの現在の状態を保存
Declare Function Api_SaveDC& Lib "gdi32" Alias "SaveDC" (ByVal hDC&)

' SaveDC関数で保存したデバイスコンテキストを復元
Declare Function Api_RestoreDC& Lib "gdi32" Alias "RestoreDC" (ByVal hDC&, ByVal
nSaveDC&)

' 論理ペンを作成
Declare Function Api_CreatePen& Lib "gdi32" Alias "CreatePen" (ByVal nPenStyle&, ByVal
nWidth&, ByVal crColor&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' 指定されたデバイスコンテキストのオブジェクトを選択
Declare Function Api_SelectObject& Lib "gdi32" Alias "SelectObject" (ByVal hDC&, ByVal
hObject&)

' 長方形の描画
Declare Function Api_Rectangle& Lib "gdi32" Alias "Rectangle" (ByVal hDC&, ByVal X1&,
ByVal Y1&, ByVal X2&, ByVal Y2&)

' 論理オブジェクトを削除し、そのオブジェクトに関連付けられていたすべてのシステムリソースを解放
Declare Function Api_DeleteObject& Lib "gdi32" Alias "DeleteObject" (ByVal hObject&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

#define PS_DASH 1                    '破線のペンを作成(ペンの幅がデバイス単位で1以下の場合のみ有効)
#define PS_DASHDOT 3                 '一点鎖線のペンを作成(ペンの幅がデバイス単位で1以下の場合のみ有効)
#define PS_DASHDOTDOT 4              '二点鎖線のペンを作成(ペンの幅がデバイス単位で1以下の場合のみ有効)
#define PS_DOT 2                     '点線のペンを作成(ペンの幅がデバイス単位で1以下の場合のみ有効)
#define PS_INSIDEFRAME 6             '塗りつぶし
#define PS_NULL 5                    '空のペンを作成。描画は行われない
#define PS_SOLID 0                   '実線のペンを作成

Var Shared Text1 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
```

```
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hDC As Long
    Var mSaveDC As Long
    Var mPen As Long
    Var Ret As Long

    'デバイスコンテキスト取得
    hDC = Api_GetDC(GethWnd)

    Cls
    '------------------------------
    mPen = Api_CreatePen(PS_SOLID, 10, RGB(255, 0, 0))     '論理ペンの作成
    Ret = Api_SelectObject(hDC, mPen)                      'オブジェクトを選択
    Ret = Api_Rectangle(hDC, 20, 15, 60, 60)               '長方形を描画
    mSaveDC = Api_SaveDC(hDC)                              '現在の状態を保存

    Text1.SetWindowText "太く赤い実線で枠を描画" & Chr$(13, 10) & "現在のDCを保存"
    Wait 300

    '------------------------------
    mPen = Api_CreatePen(PS_DOT, 0, RGB(0, 255, 0))        '論理ペンの作成
    Ret = Api_SelectObject(hDC, mPen)                      'オブジェクトを選択
    Ret = Api_Rectangle(hDC, 70, 15, 110, 60)              '長方形を描画
    Text1.SetWindowText "細い緑の点線で枠を描画"
    Wait 300

    '------------------------------
    mPen = Api_CreatePen(PS_DASHDOTDOT, 0, RGB(0, 0, 255))'論理ペンの作成
    Ret = Api_SelectObject(hDC, mPen)                      'オブジェクトを選択
    Ret = Api_Rectangle(hDC, 120, 15, 160, 60)             '長方形を描画

    Text1.SetWindowText "細い青の二点鎖線で枠を描画"
    Wait 300
    '------------------------------
    Ret = Api_RestoreDC(hDC, mSaveDC)                      '保存したデバイスコンテキストを復元
    Ret = Api_Rectangle(hDC, 170, 15, 210, 60)             '長方形を描画

    Text1.SetWindowText "SaveDCで保存したDCを復元して描画"

    '解放･削除
    Ret = Api_ReleaseDC(GetWnd, hDC)
    Ret = Api_DeleteObject(mPen)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

デバイスコンテキストの状態保存と復元(II)

SaveDC 指定のデバイスコンテキストの現在の状態を保存
RestoreDC SaveDC関数で保存したデバイスコンテキストを復元
SetTextColor デバイスコンテキストの文字色を変更
SetBkMode バックグラウンドの塗りつぶしモード設定
TextOut 文字を描画
GetDC 指定されたウィンドウのデバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放



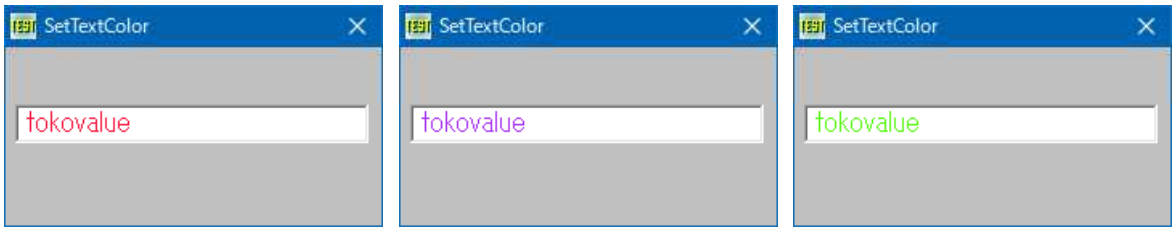

```
'=====================================
'= デバイスコンテキストの状態保存と復元(II)
'=    (SaveDC.bas)
'=====================================
#include "Windows.bi"

' 指定のデバイスコンテキストの現在の状態を保存
Declare Function Api_SaveDC& Lib "gdi32" Alias "SaveDC" (ByVal hDC&)

' SaveDC関数で保存したデバイスコンテキストを復元
Declare Function Api_RestoreDC& Lib "gdi32" Alias "RestoreDC" (ByVal hDC&, ByVal
nSaveDC&)

' デバイスコンテキストの文字色を変更
Declare Function Api_SetTextColor& Lib "gdi32" Alias "SetTextColor" (ByVal hDC&, ByVal
crColor&)

' バックグラウンドの塗りつぶしモード設定
Declare Function Api_SetBkMode& Lib "gdi32" Alias "SetBkMode" (ByVal hDC&, ByVal
iBkMode&)

' 文字を描画
Declare Function Api_TextOut& Lib "gdi32" Alias "TextOutA" (ByVal hDC&, ByVal nXStart&,
ByVal nYStart&, ByVal lpString$, ByVal cbString&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

#define TRANSPARENT 1                          '背景色を設定しない

Var Shared Button1 As Object

Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hDC As Long
    Var txt As String
    Var Ret As Long

    hDC = Api_GetDC(GethWnd)

    Ret = Api_SetBkMode(hDC, TRANSPARENT)

    txt = "DCに赤を設定"
    Ret = Api_SetTextColor(hDC, RGB(255, 0, 0))
    Ret = Api_TextOut(hDC, 10, 5, txt, Len(txt))
    Ret = Api_SaveDC(hDC)                      'DCに赤を設定したことを保存

    txt = "DCに緑を設定"
    Ret = Api_SetTextColor(hDC, RGB(0, 255, 0))
    Ret = Api_TextOut(hDC, 10, 21, txt, Len(txt))
    Ret = Api_SaveDC(hDC)                      'DCに緑を設定したことを保存
```

```
    txt = "DCに青を設定"
    Ret = Api_SetTextColor(hDC, RGB(0, 0, 255))
    Ret = Api_TextOut(hDC, 10, 37, txt, Len(txt))

    txt = "直前に保存したDCを復元"
    Ret = Api_RestoreDC(hDC, -1)
    Ret = Api_TextOut(hDC, 10, 53, txt, Len(txt))

    txt = "その前に保存したDCを復元"
    Ret = Api_RestoreDC(hDC, -1)
    Ret = Api_TextOut(hDC, 10, 69, txt, Len(txt))

    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

デバイスコンテキストの文字色を変更

TextOut 文字を描画
SetTextColor デバイスコンテキストの文字色を変更
SetBkMode バックグラウンドの塗りつぶしモード設定
GetDC 指定されたウィンドウのデバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放

例では、テキストボックス内に描画する文字の色を変えています。




```
'=====================================
'= デバイスコンテキストの文字色を変更
'=    (SetTextColor.bas)
'=====================================
#include "Windows.bi"

#define TRANSPARENT 1                       '背景色を設定しない

' 文字を描画
Declare Function Api_TextOut& Lib "gdi32" Alias "TextOutA" (ByVal hDC&, ByVal nXStart&,
ByVal nYStart&, ByVal lpString$, ByVal cbString&)

' デバイスコンテキストの文字色を変更
Declare Function Api_SetTextColor& Lib "gdi32" Alias "SetTextColor" (ByVal hDC&, ByVal
crColor&)

' バックグラウンドの塗りつぶしモード設定
Declare Function Api_SetBkMode& Lib "gdi32" Alias "SetBkMode" (ByVal hDC&, ByVal
iBkMode&)
' 指定されたウィンドウのデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)
```

```
ar Shared Text1 As Object
Var Shared Timer1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 12
Timer1.Attach GetDlgItem("Timer1")

Var Shared hDC As Long
Var Shared txt As String

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    hDC = Api_GetDC(Text1.GethWnd)
    txt = "tokovalue"

    Randomize Time
    Timer1.SetInterval 100
    Timer1.Enable -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var Ret As Long

    Ret = Api_SetTextColor(hDC, RGB(Rnd(1) * 255, Rnd(1) * 255, Rnd(1) * 255))
    Ret = Api_SetBkMode(hDC, TRANSPARENT)
    Ret = Api_TextOut(hDC, 5, 0, txt, Len(txt))
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
    Var Ret As Long

    Ret = Api_ReleaseDC(Text1.GethWnd, hDC)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

デバイスコンテキストのレイアウトを取得

FindWindow クラス名またはキャプションを与えてウィンドウハンドルを取得
GetLayout デバイスコンテキストのレイアウトを返す
SetLayout デバイスコンテキストのレイアウトを変更
GetDC デバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放

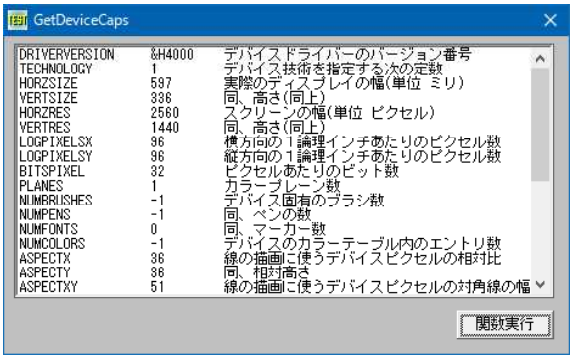
例では、メモ帳を起動後そのデバイスコンテキストにレイアウトを設定し、表示させています。


```
'=====================================
'= デバイスコンテキストのレイアウトを取得
'=    (GetLayout.bas)
'=====================================
#include "Windows.bi"

' クラス名またはキャプションを与えてウィンドウのハンドルを取得
Declare Function Api_FindWindow& Lib "user32" Alias "FindWindowA" (ByVal lpClassName$,
ByVal lpWindowName$)

' デバイスコンテキスト(DC)のレイアウトを返す
Declare Function Api_GetLayout& Lib "gdi32" Alias "GetLayout" (ByVal hDC&)

' デバイスコンテキスト(DC)のレイアウトを変更
Declare Function Api_SetLayout& Lib "gdi32" Alias "SetLayout" (ByVal hDC&, ByVal
dwLayout&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

#define LAYOUT_RTL &H1                          '既定の水平方向レイアウトを「右から左」に設定
#define LAYOUT_LTR &H2                          '既定の水平方向レイアウトを「左から右」に設定

Var Shared Text(1) As Object
Var Shared Radio(1) As Object
Var Shared Button(1) As Object

For i = 0 To 1
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1))) : Text(i).SetFontSize 14
    Radio(i).Attach GetDlgItem("Radio" & Trim$(Str$(i + 1))) : Radio(i).SetFontSize 14
    Button(i).Attach GetDlgItem("Button" & Trim$(Str$(i + 1))) : Button(i).SetFontSize 14
Next

'=====================================
'=
'=====================================
Declare Function Layout bdecl () As Integer
Function Layout()
    Layout = Val(Mid$(GetDlgRadioSelect("Radio1"), 6)) - 1
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()

    'コマンドボタンを無効に設定
    Button(0).EnableWindow 0

    'メモ帳を起動
    Shell "Notepad.exe"
End Sub

'=====================================
'=
'=====================================
```

```
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var ClassName As String
    Var Caption As String
    Var hWnd As Long
    Var hWinDC As Long
    Var Flag As Long
    Var Ret As Long

    'クラス名でウィンドウハンドルを取得
    ClassName = "Notepad"
    Caption = "無題 - メモ帳"

    hWnd = Api_FindWindow(ClassName, Caption)

    'ウィンドウハンドルを取得できたときは
    If hWnd <> 0 Then

        'デバイスコンテキストのハンドルを取得
        hWinDC = Api_GetDC(hWnd)

        'デバイスコンテキストのレイアウトを設定
        If Layout = 0 Then
            Ret = Api_SetLayout(hWinDC, LAYOUT_LTR)
        Else
            Ret = Api_SetLayout(hWinDC, LAYOUT_RTL)
        End If

        'デバイスコンテキストのレイアウトを取得
        Flag = Api_GetLayout(hWinDC)

        'レイアウトを表示
        If Flag = LAYOUT_LTR Then
            Text(1).SetWindowtext "左から右"
        Else If Flag = LAYOUT_RTL Then
            Text(1).SetWindowtext "右から左"
        End If

        'デバイスコンテキストを解放
        Ret = Api_ReleaseDC(hWnd, hWinDC)
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

デバイス固有情報の取得

GetDeviceCaps デバイス固有情報を取得
GetDC デバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放



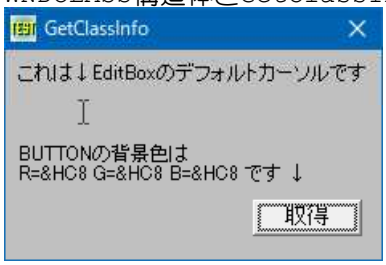

```
'========================================
'= デバイス固有情報の取得
'=    (GetDeviceCaps.bas)
'========================================
#include "Windows.bi"

' デバイス固有の情報を取得
Declare Function Api_GetDeviceCaps& Lib "gdi32" Alias "GetDeviceCaps" (ByVal hDC&, ByVal
nIndex&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

Var Shared List1 As Object
List1.Attach GetDlgItem("List1") : List1.SetFontSize 12

Var Shared cons$(30,2) As String

'========================================
'=
'========================================
Declare Sub Mainform_Start edecl ()
Sub Mainform_Start()
    For i = 0 To 30
        For j = 0 To 2
            Read cons$(i, j)
        Next j
    Next i

    data DRIVERVERSION      ,    0    ,  デバイスドライバーのバージョン番号
    data TECHNOLOGY         ,    2    ,  デバイス技術を指定する次の定数
    data HORZSIZE           ,    4    ,  実際のディスプレイの幅(単位 ミリ)
    data VERTSIZE           ,    6    ,  同、高さ(同上)
    data HORZRES            ,    8    ,  スクリーンの幅(単位 ピクセル)
    data VERTRES            ,   10    ,  同、高さ(同上)
    data LOGPIXELSX         ,   88    ,  横方向の1論理インチあたりのピクセル数
    data LOGPIXELSY         ,   90    ,  縦方向の1論理インチあたりのピクセル数
    data BITSPIXEL          ,   12    ,  ピクセルあたりのビット数
    data PLANES             ,   14    ,  カラープレーン数
    data NUMBRUSHES         ,   16    ,  デバイス固有のブラシ数
    data NUMPENS            ,   18    ,  同、ペンの数
    data NUMFONTS           ,   22    ,  同、マーカー数
    data NUMCOLORS          ,   24    ,  デバイスのカラーテーブル内のエントリ数
    data ASPECTX            ,   40    ,  線の描画に使うデバイスピクセルの相対比
    data ASPECTY            ,   42    ,  同、相対高さ
    data ASPECTXY           ,   44    ,  線の描画に使うデバイスピクセルの対角線の幅
    data PDEVICESIZE        ,   26    ,  (使用不可)
    data CLIPCAPS           ,   36    ,  クリッピング能力(戻り値は以下の定数)
    data SIZEPALETTE        ,  104    ,  システムパレット内のエントリ数
    data NUMRESERVED        ,  106    ,  同、予約エントリ数
    data COLORRES           ,  108    ,  デバイスのカラー解像度(単位 ビット/ピクセル)
    data PHYSICALWIDTH      ,  110    ,  物理的幅(単位 ピクセル)
    data PHYSICALHEIGHT     ,  111    ,  同、高さ
```

```
    data PHYSICALOFFSETX     ,   112   ,   実際に印刷可能なX方向のマージン
    data PHYSICALOFFSETY     ,   113   ,   同、Y方向マージン
    data RASTERCAPS          ,    38   ,   ラスタ能力(戻り値は以下の定数の組み合わせ)
    data CURVECAPS           ,    28   ,   デバイスがサポートする曲線描画能力
    data LINECAPS            ,    30   ,   デバイスがサポートする線分描画能力
    data POLYGONALICAPS      ,    32   ,   デバイスがサポートする多角形描画能力
    data TEXTCAPS            ,    32   ,   デバイスがサポートする文字描画能力
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hDC As Long
    Var Ret As Long

    hDC = Api_GetDC(GethWnd)

    List1.ResetContent
    For i = 0 To 30
        Ret = Api_GetDeviceCaps(hDC, Val(cons$(i,1)))
        Select Case i
            Case 0
                List1.AddString Left$(cons$(i,0) & Space$(18), 18) & "&H" & Hex$(Ret) &
Space$(10 - Len("&H" & Hex$(Ret))) & cons$(i, 2)
            Case 26, 30
                List1.AddString Left$(cons$(i,0) & Space$(18), 18) & "&H" &
Trim$(Str$(Ret)) & Space$(10 - Len("&H" & Trim$(Str$(Ret)))) & cons$(i, 2)
            Case Else
                List1.AddString Left$(cons$(i,0) & Space$(18), 18) & Trim$(Str$(Ret)) &
Space$(10 - Len(Trim$(Str$(Ret)))) & cons$(i, 2)
        End Select
    Next i
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

デフォルトカーソル描画(Ⅰ)

例ではエディットボックスのデフォルトカーソルを取得描画しています。
GetClassInfo クラス情報を取得
DrawIcon アイコンを描画
GetSysColor システムの背景色を取得
GetDC デバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放

WNDCLASS構造体とGetClassInfo



Ret = Api_DrawIcon(hDC, 38, 28, WCX.hCursor)での38,28は矢印の位置に表示したかっただけ…
クラス名
"BUTTON"　　:ボタンコントロール(ボタン・ラジオ・チェック・グループ)

```
"EDIT"          :エディットボックス
"STATIC"        :テキストボックス
"COMBOBOX"      :コンボボックス
"LISTBOX"       :リストボックス
"SCROLLBAR"     :スクロールバー

'=====================================
'= デフォルトカーソル描画(Ⅰ)
'=    (GetClassInfo.bas)
'=====================================
#include "Windows.bi"

Type WNDCLASS
    style             As Long               'ウィンドウのスタイル
    lpfnwndproc       As Long               'ウィンドウプロシージャ
    cbClsextra        As Long               '補助メモリ
    cbWndExtra2       As Long               '補助メモリ
    hInstance         As Long               'インスタンスハンドル
    hIcon             As Long               'アイコン
    hCursor           As Long               '背景ブラシ
    hbrBackground     As Long               'カーソル
    lpszMenuName      As String * 255       'メニュー名
    lpszClassName     As String * 255       'クラス名
End Type

' クラス情報を取得
Declare Function Api_GetClassInfo& Lib "user32" Alias "GetClassInfoA" (ByVal hInstance&,
ByVal lpClassName$, lpWndClass As WNDCLASS)

' アイコンを描画
Declare Function Api_DrawIcon& Lib "user32" Alias "DrawIcon" (ByVal hDC&, ByVal x&, ByVal
y&, ByVal exhIcon&)

' システムの背景色を取得
Declare Function Api_GetSysColor& Lib "user32" Alias "GetSysColor" (ByVal nIndex&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得。
その後、GDI 関数を使って、返されたデバイスコンテキスト内で描画を行える
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

Var Shared Text1 As Object
Var Shared Text2 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 12
Text2.Attach GetDlgItem("Text2") : Text2.SetFontSize 12

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var wc As WNDCLASS
    Var hDC As Long
    Var bCol As Long
    Var Ret As Long

    hDC = Api_GetDC(GethWnd)                          '構造体斎須を設定

    Ret = Api_GetClassInfo(ByVal 0, "EDIT", wc)     'クラス情報取得(EDIT)
    Text1.SetWindowText "これは↓EditBoxのデフォルトカーソルです"
    Ret = Api_DrawIcon(hDC, 38, 28, wc.hCursor)     'フォームにデフォルトカーソルを描画

    Ret = Api_GetClassInfo(ByVal 0, "BUTTON", wc)  'クラス情報取得(BUTTON)
    bCol = Api_GetSysColor(wc.hbrBackground)
    Text2.SetWindowText "BUTTONの背景色は" & Chr$(13) & "R=&&H" & hex$(bCol and &HFF) & "
G=&&H" & hex$((bCol and &HFF00) / 2 ^ 8) & " B=&&H" & hex$((bCol and &HFF0000) / 2 ^ 16) & " で
す ↓"
```

```
    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

デフォルトカーソル描画（II）

例ではエディットボックスのデフォルトカーソルを取得描画しています。
GetClassInfoEx クラス構造体から指定の項目の32ビット値を取得
DrawIcon アイコンを描画
GetSysColor システムの背景色を取得
GetDC デバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放

WNDCLASSEX構造体とGetClassInfoEx


```
'=====================================
'= デフォルトカーソル描画（II）
'=    (GetClassInfoEx.bas)
'=====================================
#include "Windows.bi"

Type WNDCLASSEX
    cbSize          As Long             '構造体サイズ
    style           As Long             'クラススタイル
    lpfnWndProc     As Long             'ウィンドウプロシージャ
    cbClsExtra      As Long             'クラス32ビット値のバイト数
    cbWndExtra      As Long             'ウィンドウ32ビット値のバイト数
    hInstance       As Long             'インスタンス
    hIcon           As Long             'アイコン
    hCursor         As Long             'カーソル
    hbrBackground   As Long             '背景ブラシ
    lpszMenuName    As Long             'メニュー
    lpszClassName   As String * 255     'クラス名
    hIconSm         As Long * 255       '小さいアイコン
End Type

' クラス構造体から指定の項目の32ビット値を取得
Declare Function Api_GetClassInfoEx& Lib "user32" Alias "GetClassInfoExA" (ByVal
hInstance&, ByVal lpClassName$, lpWndClass As WNDCLASSEX)

' アイコンを描画
Declare Function Api_DrawIcon& Lib "user32" Alias "DrawIcon" (ByVal hDC&, ByVal x&, ByVal
y&, ByVal exhIcon&)

' システムの背景色を取得
Declare Function Api_GetSysColor& Lib "user32" Alias "GetSysColor" (ByVal nIndex&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得。
その後、GDI 関数を使って、返されたデバイスコンテキスト内で描画を行える
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)
```

```
' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

Var Shared Text1 As Object
Var Shared Text2 As Object
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 12
Text2.Attach GetDlgItem("Text2") : Text2.SetFontSize 12

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var WCX As WNDCLASSEX
    Var hDC As Long
    Var bCol As Long
    Var Ret As Long

    hDC = Api_GetDC(GethWnd)

    WCX.cbSize = Len(WCX)                              '構造体 サイズを設定

    Ret = Api_GetClassInfoEx(ByVal 0, "EDIT", WCX) 'クラス情報取得(EDIT)
    Text1.SetWindowText "これは↓EditBoxのデフォルトカーソルです"
    Ret = Api_DrawIcon(hDC, 38, 28, WCX.hCursor)    'フォームにデフォルトカーソルを描画

    Ret = Api_GetClassInfoEx(ByVal 0, "BUTTON", WCX)'クラス情報取得(BUTTON)
    bCol = Api_GetSysColor(WCX.hbrBackground)
    Text2.SetWindowText "BUTTONの背景色は" & Chr$(13) & "R=&&H" & hex$(bCol and &HFF) & "
G=&&H" & hex$((bCol and &HFF00) / 2 ^ 8) & " B=&&H" & hex$((bCol and &HFF0000) / 2 ^ 16) & " で
す ↓"

    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

デフォルトのE-Mailアドレスを取得

RegOpenKeyEx レジストリのキーのハンドルを確保
RegQueryValueEx レジストリの値を取得
RegCloseKey レジストリのハンドルを解放

注)OSにより、サブキーの位置が異なる場合があります。

```
'=====================================
'= デフォルトのE-Mailアドレスを取得
'=    (GetDefaultEmailAdd.bas)
'=====================================
#include "Windows.bi"

#define HKEY_CURRENT_USER -2147483647      '現在Windowsにログインしているユーザーの情報
```

```
#define REG_SZ 1                                'ヌル終端文字列
#define ERROR_SUCCESS &H0                       '正常終了の戻り値を示す
#define STANDARD_RIGHTS_READ &H20000            '(READ_CONTROL)オブジェクトのセキュリティ記述子の読
                                                み取り許可
#define KEY_QUERY_VALUE &H1                     'サブキーデータを問い合わせるためのアクセス権
#define KEY_ENUMERATE_SUB_KEYS &H8              'サブキーの列挙を許可
#define KEY_NOTIFY &H10                         '変更の通知を許可
#define SYNCHRONIZE &H100000                    '同期をとる

' レジストリのキーのハンドルを確保
Declare Function Api_RegOpenKeyEx& Lib "advapi32" Alias "RegOpenKeyExA" (ByVal hKey&,
ByVal lpSubKey$, ByVal ulOptions&, ByVal samDesired&, phkResult&)

' レジストリの値を取得
Declare Function Api_RegQueryValueEx& Lib "advapi32" Alias "RegQueryValueExA" (ByVal
hKey&, ByVal lpvName$, ByVal lpReserved&, ByVal lpType&, ByVal lpData$, lpcbData&)

' レジストリのハンドルを解放
Declare Function Api_RegCloseKey& Lib "advapi32" Alias "RegCloseKey" (ByVal hKey&)

Var Shared Text1 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Function TrimNull(startstr As String) As String
Function TrimNull(startstr As String) As String
    TrimNull = Left$(startstr, InStr(startstr, Chr$(0)) - 1)
End Function

'=====================================
'=
'=====================================
Declare Function OpenRegKey(hKey As Long, lpSubKey As String) As Long
Function OpenRegKey(hKey As Long, lpSubKey As String) As Long
    Var hSubKey As Long

    If Api_RegOpenKeyEx(hKey, lpSubKey, 0, STANDARD_RIGHTS_READ Or KEY_QUERY_VALUE Or
KEY_ENUMERATE_SUB_KEYS Or KEY_NOTIFY And Not SYNCHRONIZE, hSubKey) = ERROR_SUCCESS Then
        OpenRegKey = hSubKey
    End If
End Function

'=====================================
'=
'=====================================
Declare Function GetRegValue(hSubKey As Long, sKeyName As String) As String
Function GetRegValue(hSubKey As Long, sKeyName As String) As String
    Var lpValue As String
    Var lpcbData As Long

    If hSubKey <> 0 Then
        lpValue = Space$(260)
        lpcbData = Len(lpValue)
        If Api_RegQueryValueEx(hSubKey, sKeyName, 0, 0, lpValue, lpcbData) =
ERROR_SUCCESS Then
            GetRegValue = TrimNull(lpValue)
        End If
    End If
End Function

'=====================================
'=
'=====================================
Declare Function GetDefaultAccount() As String
Function GetDefaultAccount() As String
```

```
    Var hKey As Long
    Var sKey As String
    Var Ret As Long

    sKey = "Software¥Microsoft¥Internet Account Manager"
    hKey = OpenRegKey(HKEY_CURRENT_USER, sKey)

    If hKey <> 0 Then
        GetDefaultAccount = GetRegValue(hKey, "Default Mail Account")
        Ret = Api_RegCloseKey(hKey)
   End If
End Function

'=====================================
'=
'=====================================
Declare Function GetDefaultEmailAddress() As String
Function GetDefaultEmailAddress() As String
    Var sAccount As String
    Var sDefAddress As String
    Var hKey As Long
    Var sKey As String
    Var tmp As String
    Var Ret As Long

    sAccount = GetDefaultAccount()

    If Len(sAccount) <> 0 Then
        sKey = "Software¥Microsoft¥Internet Account Manager"
        hKey = OpenRegKey(HKEY_CURRENT_USER, sKey & "¥Accounts¥" & sAccount)
        If hKey <> 0 Then
            tmp = GetRegValue(hKey, "SMTP Email Address")
            If Len(tmp) > 0 Then
                GetDefaultEmailAddress = tmp
                Exit Function
            Else
                tmp = GetRegValue(hKey, "SMTP Reply To Email Address")
                If Len(tmp) > 0 Then
                    GetDefaultEmailAddress = tmp
                    Exit Function
                Else
                    tmp = GetRegValue(hKey, "POP3 Email Address")
                    If Len(tmp) > 0 Then
                        GetDefaultEmailAddress = tmp
                        Exit Function
                    Else
                        tmp = GetRegValue(hKey, "POP3 Reply To Email Address")
                        If Len(tmp) > 0 Then
                            GetDefaultEmailAddress = tmp
                            Exit Function
                        Else
                            GetDefaultEmailAddress = ""
                        End If
                    End If
                End If
            End If
            Ret = Api_RegCloseKey(hKey)
        End If
    End If
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Text1.SetWindowText GetDefaultEmailAddress
End Sub
```

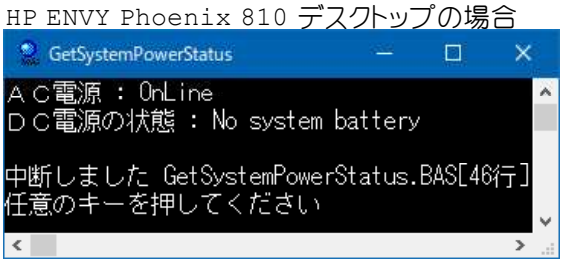
```
'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

電源(AC･DC)の状態を取得

電源の状態を取得します。
GetSystemPowerStatus 電源の状態を取得

NEC VL100/2(AC専用)の場合

HP ENVY Phoenix 810 デスクトップの場合



NEC LL750D(Note)AC電源使用時

NEC LL750D(Note)DC電源使用時(満充電)

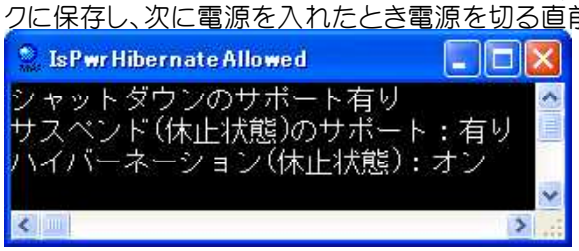

```
'=====================================
'= 電源(AC･DC)の状態を取得
'=    (GetSystemPowerStatus.bas)
'=====================================

Type SYSTEM_POWER_STATUS
    ACLineStatus            As Byte         'ACパワーの状態を示す定数の組み合わせ
    BatteryFlag             As Byte         'バッテリチャージの状態を表す定数の組み合わせ
    BatteryLifePercent      As Byte         'バッテリの残り容量のパーセント(0～100)
    Reserved1               As Byte         '常に0
    BatteryLifeTime         As Long         'バッテリーの残り秒数
    BatteryFullLifeTime     As Long         'フルチャージ時の残り秒数
End Type

' 電源の状態を取得
Declare Function Api_GetSystemPowerStatus& Lib "kernel32" Alias "GetSystemPowerStatus"
(lpSystemPowerStatus As SYSTEM_POWER_STATUS)

var SPS As SYSTEM_POWER_STATUS
var Ret As Long

Ret = Api_GetSystemPowerStatus(SPS)

Select Case SPS.ACLineStatus
    Case 0
        Print "AC電源 : OffLine"
    Case 1
        Print "AC電源 : OnLine"
    Case 2
        Print "AC電源 : 判別不能"
End Select

Select Case SPS.BatteryFlag
    Case 1
```

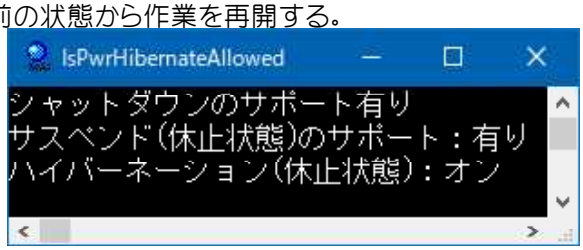
```
        Print "DC電源の状態 : High"
    Case 2
        Print "DC電源の状態 : Low"
    Case 4
        Print "DC電源の状態 : Critical"
    Case 8
        Print "DC電源の状態 : Charging"
    Case 128
        Print "DC電源の状態 : No system battery"
    Case 255
        Print "DC電源の状態 : Unknown Status"
End Select

Stop
End
```

電源オプションのサポート状態を取得

IsPwrShutdownAllowed シャットダウンをサポートするかどうかを取得
IsPwrSuspendAllowed サスペンド(スタンバイ)をサポートするかどうかを取得
IsPwrHibernateAllowed ハイバネーション(休止状態)かどうかを取得

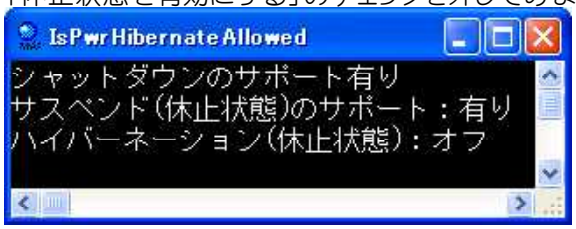
★実行中プログラムの終了処理などを正しく行ないシステムの電源を遮断できる状態にする。
★サスペンド(suspend)別名をスタンバイともいい、コンピュータの電源を切る直前の状態をメモリに保存し、次に電源を入れたとき電源を切る直前の状態から作業を再開する。
★ハイバーネーション(hibernation)別名を休止状態といい、コンピュータの電源を切る直前の状態をハードディスクに保存し、次に電源を入れたとき電源を切る直前の状態から作業を再開する。

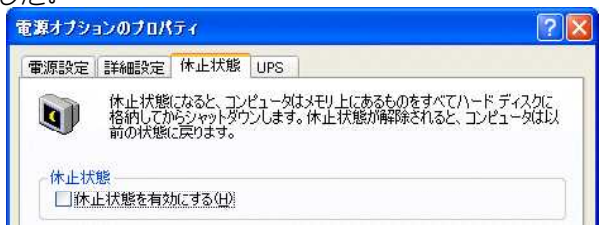




Windows10
デスクトップ

「休止状態を有効にする」のチェックを外してみました。





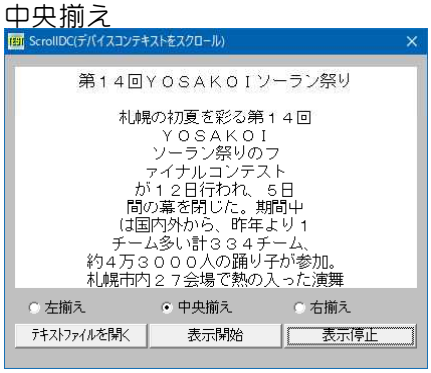

```
'=======================================
'= 電源オプションのサポート状態を取得
'=    (IsPwrHibernateAllowed.bas)
'=======================================

' シャットダウンのサポートするかどうかを取得
Declare Function Api_IsPwrShutdownAllowed& Lib "Powrprof" Alias "IsPwrShutdownAllowed"
()

' 休止状態のサポートするかどうかを取得
Declare Function Api_IsPwrSuspEndAllowed& Lib "Powrprof" Alias "IsPwrSuspEndAllowed" ()

' ハイバネーションのサポートするかどうかを取得
Declare Function Api_IsPwrHibernateAllowed& Lib "Powrprof" Alias
"IsPwrHibernateAllowed" ()

If Api_IsPwrShutdownAllowed <> 0 Then
    Print "シャットダウンのサポート有り"
Else
    Print "シャットダウンのサポート無し"
End If
```

```
If Api_IsPwrSuspEndAllowed <> 0 Then
    Print "休止状態のサポート:有り"
Else
    Print "休止状態のサポート:無し"
End If

If Api_IsPwrHibernateAllowed <> 0 Then
    Print "ハイバーネーション(休止状態):オン"
Else
    Print "ハイバーネーション(休止状態):オフ"
End If

Stop
End
```

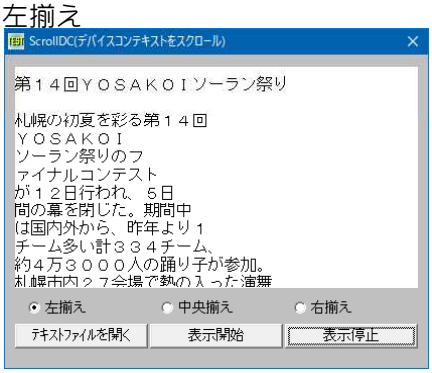
電光掲示板(デバイスコンテキストのスクロール)

ピクチャボックスに描画した文字列を周期的に1ドットずつ上に移動させます。文字列はテキストファイルから読み込んでいます。
ラジオボタンでAlignmentを設定できます。
GetTickCount システムが起動してからの経過時間を取得
SetRect RECT構造体の値を設定
OffsetRect 矩形領域の補正
ScrollDC デバイスコンテキストをスクロール
DrawText 文字列を指定領域に出力

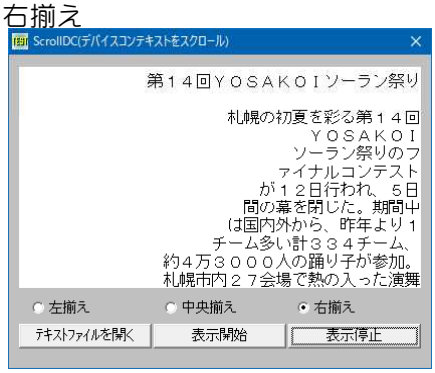
中央揃え



左揃え



右揃え




```
'=======================================
'= 電光掲示板(デバイスコンテキストのスクロール)
'=    (ScrollDC.bas)
'=======================================
#include "Windows.bi"
Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

' システムが起動してからの経過時間を取得
Declare Function Api_GetTickCount& Lib "kernel32" Alias "GetTickCount" ()

' RECT構造体の値を設定
Declare Function Api_SetRect& Lib "user32" Alias "SetRect" (lpRect As RECT, ByVal X1&,
ByVal Y1&, ByVal X2&, ByVal Y2&)

' 矩形領域の補正
Declare Function Api_OffsetRect& Lib "user32" Alias "OffsetRect" (lpRect As RECT, ByVal
x&, ByVal y&)

' デバイスコンテキストをスクロール
Declare Function Api_ScrollDC& Lib "user32" Alias "ScrollDC" (ByVal hDC&, ByVal dx&,
ByVal dy&, lprcScroll As RECT, lprcClip As RECT, ByVal hrgnUpdate&, lprcUpdate As RECT)
```

```
' 文字列を指定領域に出力
Declare Function Api_DrawText& Lib "user32" Alias "DrawTextA" (ByVal hDC&, ByVal lpStr$,
ByVal nCount&, lpRect As RECT, ByVal wFormat&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

Var Shared Button1 As Object
Var Shared Button2 As Object
Var Shared Button3 As Object
Var Shared Picture1 As Object

Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14
Button3.Attach GetDlgItem("Button3") : Button3.SetFontSize 14
Picture1.Attach GetDlgItem("Picture1")

Var Shared txtLine(100) As String
Var Shared Max As Integer
Var Shared Scrolling As Integer
Var Shared t As Long
Var Shared Index As Long
Var Shared rText As RECT
Var Shared rClip As RECT
Var Shared rUpdate As RECT
Var Shared hDC As Long
Var Shared CrLf As String

'=====================================
'=
'=====================================
Declare Function Alignment bdecl () As Integer
Function Alignment()
    Alignment = Val(Mid$(GetDlgRadioSelect("Radio1"), 6)) - 1
End Function

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var Ret As Long

    CrLf = Chr$(13, 10)

    'ピクチャボックスDC取得
    hDC = Api_GetDC(Picture1.GethWnd)

    '矩形範囲を設定
    Ret = Api_SetRect(rClip, 0, 1, Picture1.GetWidth, Picture1.GetHeight)
    Ret = Api_SetRect(rText, 0, Picture1.GetHeight, Picture1.GetWidth,
Picture1.GetHeight + GetFontSize)
End Sub

'=====================================
'=
'=====================================
Declare Sub Scroll edecl ()
Sub Scroll()
    Var txt As String
    Var Ret As Long

    do
        '周期的な枠
        If Api_GetTickCount - t > 25 Then
            t = Api_GetTickCount
           If rText.Bottom < Picture1.GetHeight Then
```

```
                Ret = Api_OffsetRect(rText, 0, GetFontSize)

                If Alignment = &H1 Then
                    txt = Trim$(txtLine(Index))
                Else
                    txt = txtLine(Index)
                End If

                Index = Index + 1
            End If

            Ret = Api_DrawText(hDC, txt, Len(txt), rText, Alignment)
            Ret = Api_OffsetRect(rText, 0, -1)
            Ret = Api_ScrollDC(hDC, 0, -1, rClip, rClip, 0, rUpdate)

            Picture1.Line(0, Picture1.GetHeight - 1) - (Picture1.GetWidth,
Picture1.GetHeight - 1) , , 15
        End If

        CallEvent
    Loop Until Scrolling = 0 Or Index >= Max

    Button1.EnableWindow -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var FileName As String
    Var sFile As String
    Var FF As Integer

    FF = FreeFile
    FileName = WinOpenDlg("ファイルのオープン", "*.txt", "テキスト(*.txt)", 0)
    If FileName <> Chr$(&H1B) Then
        Open FileName For Input As #FF
    Else
        Close #FF
        Exit Sub
    End If

    Index = 0

    While Not eof(FF)
        Line Input #FF, sFile
        Index = Index + 1
        If Index > 100 Then Max = Index : Close #FF : Exit Sub
        txtLine(Index) = sFile
    Wend

    Close #FF
    Max = Index
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Button1.EnableWindow 0
    Scrolling = -1
    Index = 0
    Scroll
End Sub

'=====================================
'=
'=====================================
```

```
Declare Sub Button3_on edecl ()
Sub Button3_on()
    Scrolling = 0
    Button2.EnableWindow -1
End Sub

'=======================================
'=
'=======================================
Declare Sub MainForm_QueryClose edecl (Cancel%, Mode%)
Sub MainForm_QueryClose(Cancel%, Mode%)
    Scrolling = 0
    Ret = Api_ReleaseDC(Picture1.GethWnd, hDC)
    End
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

電卓(F-BASICのSAMPLEより)

F-BASIC63のSAMPLEです。短いプログラムに納得の手法が見てとれます。

```
declare sub Button1_on edecl ()
sub Button1_on()
    処理
end sub
```

これをボタンの数だけ並べると、大変ですヨネ••
プロパティで数字Buttonのイベントを全てButtonNum_onとし、クリックされたButtonのTextを取得しval関数で数値に変換しています。
加減乗除およびクリアButtonのイベントは全てButtonOpl_onとし、クリックされたButtonのTextを取得し加減乗除・クリアを実行しています。

フォームの体裁のみ変更しています。

```
'=======================================
'= 電卓のサンプルプログラム(富士通ミドルウェアSampleより)
'=======================================
#include "Windows.bi"

Var Shared Text1 As Object
Var Shared FF
Var Shared Num1
Var Shared Num2
FF = 0

Text1.Attach GetDlgItem("Text1")
Text1.SetWindowText "0"

'=======================================
'=
'=======================================
```

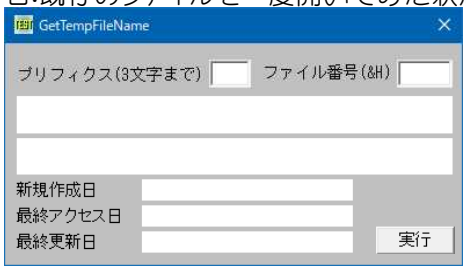
```
Declare Sub ButtonOpl_on edecl ()
Sub ButtonOpl_on ()
    Var Button As Object
    Static Opl$
    Button.Attach GETFOCUS
    FF = 1
    Select Case Button.GetWindowText
        Case "C"
            Text1.SetWindowText "0"
        Case "AC"
            Text1.SetWindowText "0"
            FF = 0
        Case "+", "-", "*",  "/"
            Select Case Opl$
                Case "+"
                    Num2 = Val(Text1.GetWindowText)
                    Num1 = Num1 + Num2
                    Text1.SetWindowText Mid$(Str$(Num1), 2)
                Case "-"
                    Num2 = Val(Text1.GetWindowText)
                    Num1 = Num1 - Num2
                    Text1.SetWindowText Mid$(Str$(Num1), 2)
                Case "*"
                    Num2 = Val(Text1.GetWindowText)
                    Num1 = Num1 * Num2
                    Text1.SetWindowText Mid$(Str$(Num1), 2)
                Case "/"
                    Num2 = Val(Text1.GetWindowText)
                    Num1 = Num1 / Num2
                    Text1.SetWindowText Mid$(Str$(Num1), 2)
            End Select
            Opl$ = Button.GetWindowText
        Case "="
            Num2 = Val(Text1.GetWindowText)
            Select Case Opl$
                Case "+"
                    Num1 = Num1 + Num2
                Case "-"
                    Num1 = Num1 - Num2
                Case "*"
                    Num1 = Num1 * Num2
                Case "/"
                    Num1 = Num1 / Num2
            End Select
            Text1.SetWindowText Mid$(Str$(Num1), 2)
            Opl$ = ""
    End Select
End Sub

'=====================================
'=
'=====================================
Declare Sub ButtonNum_on edecl ()
Sub ButtonNum_on ()
    Var Button As Object
    Button.Attach GETFOCUS
    BNum$ = Button.GetWindowText
    Num$ = Text1.GetWindowText
    If Num$ = "0" Then
        Text1.SetWindowText BNum$
        FF = 0
    Else If FF = 1 Then
        Num1 = Val(Num$)
        FF = 0
        Text1.SetWindowText BNum$
    Else
        Text1.SetWindowText Num$ + BNum$
    End If
End Sub
```

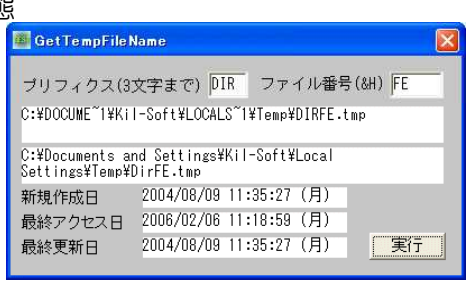
```
'=======================================
'=
'=======================================
Declare Sub Copy_on edecl ()
Sub Copy_on()
    SetCbText Text1.GetWindowText
End Sub

'=======================================
'=
'=======================================
Declare Sub Exit_on edecl ()
Sub Exit_on()
    End
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop : End
```

テンポラリフォルダとファイル名

テンポラリフォルダのファイル取得および作成
FindClose ファイルのハンドルを閉じる
CreateFile ファイルのハンドルを取得
GetFileTime ファイルのタイムスタンプを取得
FileTimeToSystemTime システムファイルタイムに変換
FileTimeToLocalFileTime ローカルタイムに変換
GetTempPath 一時フォルダ(テンポラリフォルダ)を取得
GetTempFileName 一時ファイル名取得(作成)
GetLongPathName ロングパス名を取得

ユニーク(一意)な名前を持つ空の一時ファイルを作成します。3文字までのプリフィクス(3文字以上の場合は先頭の3文字までが有効)と、システムが作成する最大4桁の16進数を加え、tmpという拡張子を持つファイル名が作成されます。例:**xxxyyyy.tmp**(xxx文字列、yyyy16進数文字列)
右:既存のファイルを一度開いてみた状態





左:テンポラリファイルは、3文字までのプリフィクスと16進数のテンポラリ番号で表される。
右:プリフィクスに**abc**、ファイル番号に**0**を入力した場合新たなファイルが作成されます。

```
'=======================================
'= テンポラリフォルダとファイル名
'=    (GetTempFileName.bas)
'=======================================
#include "Windows.bi"
```

```
Type SYSTEMTIME
    wYear             As Integer            '年
    wMonth            As Integer            '月     (1月＝1、2月＝2･･･)
    wDayOfWeek        As Integer            '曜日   (日曜＝0、月曜＝1･･)
    wDay              As Integer            '日
    wHour             As Integer            '時
    wMinute           As Integer            '分
    wSecond           As Integer            '秒
    wMilliseconds     As Integer            'ミリ秒
End Type

Type FILETIME
    dwLowDateTime     As Long               '下位32ビット値
    dwHighDateTime    As Long               '上位32ビット値
End Type

Type SECURITY_ATTRIBUTES
    nLength                  As Long
    lpSecurityDescriptor     As Long
    bInheritHandle           As Long
End Type

#define GENERIC_READ -2147483648           '&H80000000
#define GENERIC_WRITE &H40000000           '
#define FILE_SHARE_READ &H1                '後続のオープン操作で読み取りアクセスが要求された場合、そのオープンを許可
#define FILE_SHARE_WRITE &H2               '後続のオープン操作で書き込みアクセスが要求された場合、そのオープンを許可
#define CREATE_NEW 1                       '
#define CREATE_ALWAYS 2                    '
#define OPEN_EXISTING 3                    'ファイルをオープンする
#define OPEN_ALWAYS 4                      '
#define TRUNCATE_EXISTING 5                '
#define FILE_ATTRIBUTE_ARCHIVE &H20        'アーカイブ属性を示す定数
#define FILE_ATTRIBUTE_HIDDEN &H2          '隠しファイル属性
#define FILE_ATTRIBUTE_NORMAL &H80         '他のファイル属性を持たない
#define FILE_ATTRIBUTE_READONLY &H1        '読み込み専用属性
#define FILE_ATTRIBUTE_SYSTEM &H4          'システムファイル属性
#define FILE_ATTRIBUTE_TEMPORARY &H100     '一時ファイル属性を示す定数
#define FILE_FLAG_WRITE_THROUGH -2147483648      'キャッシュに書き込まれたデータをそのまま直接ディスクに書き込むようにする
#define FILE_FLAG_OVERLAPPED &H40000000    '時間のかかる処理に対してReadFile関数やWriteFile関数でERROR_IO_PENDING拡張エラー･
#define FILE_FLAG_NO_BUFFERING &H20000000 'システムキャッシュを使用せずにファイルをオープンするよう指定
#define FILE_FLAG_RANDOM_ACCESS &H10000000'ファイルにランダムアクセスすることをシステムに示す
#define FILE_FLAG_SEQUENTIAL_SCAN &H8000000      'ファイルにｼｰｹﾝｼｬﾙｱｸｾｽすることをシステムに示す
#define FILE_FLAG_DELETE_ON_CLOSE &H4000000      'そのファイルを指すすべてのファイルのハンドルがクローズされた時に、そのファイルを
#define FILE_FLAG_POSIX_SEMANTICS &H1000000      'POSIXの規則に従ってファイルにアクセス

' ファイルのハンドルを閉じる
Declare Function Api_FindClose& Lib "kernel32" Alias "FindClose" (ByVal hFindFile&)

' ファイルのハンドルをGET
Declare Function Api_CreateFile& Lib "kernel32" Alias "CreateFileA" (ByVal lpFileName$,
ByVal dwDesiredAccess&, ByVal dwShareMode&, ByVal lpSecurityAttributes&, ByVal
dwCreationDisposition&, ByVal dwFlagsAndAttributes&, ByVal hTemplateFile&)

' ファイルスタンプを取得する
Declare Function Api_GetFileTime& Lib "kernel32" Alias "GetFileTime" (ByVal hFile&,
lpCreationTime As FILETIME, lpLastAccessTime As FILETIME, lpLastWriteTime As FILETIME)

' システムファイルタイムに変換
Declare Function Api_FileTimeToSystemTime& Lib "kernel32" Alias "FileTimeToSystemTime"
(lpFileTime As FILETIME, lpSystemTime As SYSTEMTIME)

' ローカルタイムに変換
Declare Function Api_FileTimeToLocalFileTime& Lib "kernel32" Alias
"FileTimeToLocalFileTime" (lpFileTime As FILETIME, lpLocalFileTime As FILETIME)
```

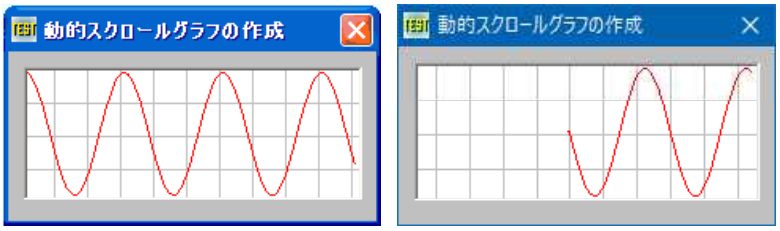
```
' 一時フォルダ(テンポラリフォルダ)を取得
Declare Function Api_GetTempPath& Lib "Kernel32" Alias "GetTempPathA" (ByVal
nBufferLength&, ByVal lpBuffer$)

' 一時ファイル名取得(作成)
Declare Function Api_GetTempFileName& Lib "kernel32" Alias "GetTempFileNameA" (ByVal
lpszPath$, ByVal lpPrefixString$, ByVal wUnique&, ByVal lpTempFileName$)

' ロングパス名を取得
Declare Function Api_GetLongPathName& Lib "Kernel32" Alias "GetLongPathNameA" (ByVal
lpszShortPath$, ByVal lpszLongPath$, ByVal cchBuffer&)

Var Shared Text(9) As Object
Var Shared Edit1 As Object
Var Shared Edit2 As Object

For i = 0 To 9
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1))) : Text(i).SetFontSize 14
Next
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Edit2.Attach GetDlgItem("Edit2") : Edit2.SetFontSize 14

'=====================================
'=
'=====================================
Declare Function WeekDay(Str As Integer) As String
Function WeekDay(Str As Integer) As String
    Select Case Str
        Case 0
            WeekDay = "日"
        Case 1
            WeekDay = "月"
        Case 2
            WeekDay = "火"
        Case 3
            WeekDay = "水"
        Case 4
            WeekDay = "木"
        Case 5
            WeekDay = "金"
        Case 6
            WeekDay = "土"
    End Select
End Function

'=====================================
'=
'=====================================
Declare Function LongPathName(strShortPathName As String) As String
Function LongPathName(strShortPathName As String) As String
    Var Buffer As String
    Var Ret As Long

    Buffer = String$(256, Chr$(0))
    On Error Goto *LongPathName_Error

    Ret = Api_GetLongPathName(strShortPathName, Buffer, Len(Buffer))
    On Error Goto 0

    ' 不正パスはNullを返す
    If Ret > 0 Then
        LongPathName = Left$(Buffer, InStr(Buffer, Chr$(0)) - 1)
    Else
        LongPathName = Chr$(0)
    End If
    Exit Function

'GetLongPathName未実装対策(For Windows95,WindowsNT4.0)
*LongPathName_Error
    LongPathName = strShortPathName
```

```
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var lpCreationTime As FILETIME
    Var lpLastAccessTime As FILETIME
    Var lpLastWriteTime As FILETIME
    Var lpLocalFileTime As FILETIME
    Var lpSystemTime As SYSTEMTIME
    Var Buffer As String * 256                  '文字列バッファ
    Var Prefix As String                        'プリフィクス(3文字まで)
    Var FileNo As Long                          'テンポラリファイル番号
    Var TempP As String                         'テンポラリパス
    Var TempN As String                         'テンポラリファイル名
    Var TempL As String
    Var RetVal As Long
    Var Ret As Long

    'プリフィクス取得
    Prefix = Edit1.GetWindowText

    'テンポラリファイル番号
    FileNo = Val("&H" & EDit2.GetWindowText)

    'テンポラリパス取得
    Ret = Api_GetTempPath(256, Buffer)
    TempP = Left$(Buffer, Ret)

    'テンポラリファイル名取得
    Ret = Api_GetTempFileName(TempP, Prefix, FileNo, Buffer)
    TempN = Left$(Buffer, InStr(Buffer, Chr$(0)) - 1)
    Text(0).SetWindowText TempN

    TempL = LongPathName(TempN)
    Text(1).SetWindowText TempL

   'ファイルのハンドル取得する
    Ret = Api_CreateFile(TempL, GENERIC_READ, FILE_SHARE_READ, 0, OPEN_EXISTING,
FILE_ATTRIBUTE_NORMAL, 0)

    If Ret <> -1 Then

        'ファイルスタンプを取得する
        RetVal = Api_GetFileTime(Ret, lpCreationTime, lpLastAccessTime, lpLastWriteTime)

        If RetVal <> 0 Then

            'ファイルの新規作成日を取得
            RetVal = Api_FileTimeToLocalFileTime(lpCreationTime, lpLocalFileTime)

            'システムタイムに変換
            RetVal = Api_FileTimeToSystemTime(lpLocalFileTime, lpSystemTime)

            If RetVal <> 0 Then
                ymd$ = Trim$(Str$(lpSystemTime.wYear)) & "/" & Right$(Str$(100 +
lpSystemTime.wMonth), 2) & "/" & Right$(Str$(100 + lpSystemTime.wDay), 2) & " "
                hms$ = Right$(Str$(100 + lpSystemTime.wHour), 2) & ":" & Right$(Str$(100 +
lpSystemTime.wMinute), 2) & ":" & Right$(Str$(100 + lpSystemTime.wSecond), 2) & " "
                week$ = "(" & WeekDay(lpSystemTime.wDayOfWeek) & ")"
                Text(2).SetWindowText ymd$ & hms$ & week$
            End If

            '最終アクセス日を取得
            RetVal = Api_FileTimeToLocalFileTime(lpLastAccessTime, lpLocalFileTime)

            'システムタイムに変換
            RetVal = Api_FileTimeToSystemTime(lpLocalFileTime, lpSystemTime)
```

```
            If RetVal <> 0 Then
                ymd$ = Trim$(Str$(lpSystemTime.wYear)) & "/" & Right$(Str$(100 +
lpSystemTime.wMonth), 2) & "/" & Right$(Str$(100 + lpSystemTime.wDay), 2) & " "
                hms$ = Right$(Str$(100 + lpSystemTime.wHour), 2) & ":" & Right$(Str$(100 +
lpSystemTime.wMinute), 2) & ":" & Right$(Str$(100 + lpSystemTime.wSecond), 2) & " "
                week$ = "(" & WeekDay(lpSystemTime.wDayOfWeek) & ")"
                Text(3).SetWindowText ymd$ & hms$ & week$
            End If

            'ファイルの最終更新日を取得
            RetVal = Api_FileTimeToLocalFileTime(lpLastWriteTime, lpLocalFileTime)

            'システムタイムに変換
            RetVal = Api_FileTimeToSystemTime(lpLocalFileTime, lpSystemTime)

            If RetVal <> 0 Then
                ymd$ = Trim$(Str$(lpSystemTime.wYear)) & "/" & Right$(Str$(100 +
lpSystemTime.wMonth), 2) & "/" & Right$(Str$(100 + lpSystemTime.wDay), 2) & " "
                hms$ = Right$(Str$(100 + lpSystemTime.wHour), 2) & ":" & Right$(Str$(100 +
lpSystemTime.wMinute), 2) & ":" & Right$(Str$(100 + lpSystemTime.wSecond), 2) & " "
                week$ = "(" & WeekDay(lpSystemTime.wDayOfWeek) & ")"
                Text(4).SetWindowText ymd$ & hms$ & week$
            End If
        End If
    End If

    Ret = Api_FindClose(Ret)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

動的スクロールグラフの作成

スクロールするサインウェーウェーブを描画します。
CreateCompatibleDC 関連するデバイスと互換性のあるデバイスコンテキストを作成
CreateCompatibleBitmap デバイスコンテキストと互換性のあるビットマップを作成
SelectObject 指定されたデバイスコンテキストのオブジェクトを選択
CreatePen 論理ペンを作成
LineTo 現在の位置から終点までを直線で描画
MoveToEx 現在位置を受け取るバッファを参照で指定
BitBlt 指定された長方形内の各ピクセルの色データをコピー
GetDC デバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放



参照
http://support.microsoft.com/kb/183333/ja

```
'=====================================
'= 動的スクロールグラフの作成
'=    (CreateCompatibleBitmap.bas)
'=    参照 http://support.microsoft.com/kb/183333/ja
'=====================================
```

```
#include "Windows.bi"

' 指定されたデバイスコンテキストに関連するデバイスと互換性のあるメモリデバイスコンテキストを作成
Declare Function Api_CreateCompatibleDC& Lib "gdi32" Alias "CreateCompatibleDC" (ByVal
hDC&)

' デバイスコンテキストと互換性のあるビットマップを作成
Declare Function Api_CreateCompatibleBitmap& Lib "gdi32" Alias "CreateCompatibleBitmap"
(ByVal hDC&, ByVal nWidth&, ByVal nHeight&)

' 指定されたデバイスコンテキストのオブジェクトを選択
Declare Function Api_SelectObject& Lib "gdi32" Alias "SelectObject" (ByVal hDC&, ByVal
hObject&)

' 論理ペンを作成
Declare Function Api_CreatePen& Lib "gdi32" Alias "CreatePen" (ByVal nPenStyle&, ByVal
nWidth&, ByVal crColor&)

' 現在の位置から終点までを直線で描画
Declare Function Api_LineTo& Lib "gdi32" Alias "LineTo" (ByVal hDC&, ByVal x&, ByVal y&)

' 現在位置を受け取るバッファを参照で指定
Declare Function Api_MoveToEx& Lib "gdi32" Alias "MoveToEx" (ByVal hDC&, ByVal x&, ByVal
y&, ByVal lpPoint&)

' ビットブロック転送を行う。コピー元からコピー先のデバイスコンテキストへ、指定された長方形内の各ピクセルの色
データをコピー
Declare Function Api_BitBlt& Lib "gdi32" Alias "BitBlt" (ByVal hDestDC&, ByVal X&, ByVal
Y&, ByVal nWidth&, ByVal nHeight&, ByVal hSrcDC&, ByVal xSrc&, ByVal ySrc&, ByVal dwRop&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

#define SRCCOPY &HCC0020                    'そのまま転送
#define PS_SOLID 0                          '実線のペンを作成

#define pWidth 210                          'ピクチャボックス幅
#define pHeight 84                          'ピクチャボックス高さ
#define pHeightHalf 40                      'Y軸（高さの半分）
#define pGrid 21                            'グリッド幅、高さ
#define tInterval 10                        'タイマーインターバル
#define vbGray &HC0C0C0                     '灰色のカラーコード
#define vbRed &H0000FF                      '赤のカラーコード

Var Shared counter As Long                  'カウンター
Var Shared oldY As Long                     '元のY座標
Var Shared hDCh As Long                     '
Var Shared phDC As Long                     'ピクチャボックスのデバイスコンテキスト
Var Shared hPenBlack As Long                '黒ペン
Var Shared hPenRed As Long                  '赤ペン

Var Shared Picture1 As Object
Var Shared Timer1 As Object

Picture1.Attach GetDlgItem("Picture1")
Timer1.Attach GetDlgItem("Timer1")

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var hBmp As Long
    Var i As Integer
    Var Ret As Long

    phDC = Api_GetDC(Picture1.GethWnd)
```

```
    Picture1.MoveWindow 10, 10
    Picture1.SetWindowSize 215, 85

    counter = 0
    hDCh = Api_CreateCompatibleDC(phDC)
    hBmp = Api_CreateCompatibleBitmap(phDC, pWidth, pHeight)
    Ret = Api_SelectObject(hDCh, hBmp)
    hPenBlack = Api_CreatePen(PS_SOLID, 0, vbGray)
    hPenRed = Api_CreatePen(PS_SOLID, 0, vbRed)
    Ret = Api_SelectObject(hDCh, hPenBlack)

    '水平グリッドライン
    For i = pGrid To pHeight - 1 Step pGrid
        Picture1.Line (0, i)-(pWidth, i),, 14
    Next

    '垂直グリッドライン
    For i = pGrid - (counter Mod pGrid) To pWidth - 1 Step pGrid
        Picture1.Line (i, 0)-(i, pHeight - 1),, 14
    Next

    Ret = Api_BitBlt(hDCh, 0, 0, pWidth, pHeight, phDC, 0, 0, SRCCOPY)
    Timer1.SetInterval 10
    Timer1.Enable -1
    oldY = pHeightHalf
End Sub

'=======================================
'=
'=======================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var i As Integer

    Ret = Api_BitBlt(hDCh, 0, 0, pWidth - 1, pHeight, hDCh, 1, 0, SRCCOPY)

    If counter Mod pGrid = 0 Then
        Ret = Api_MoveToEx(hDCh, pWidth - 2, 0, 0)
        Ret = Api_LineTo(hDCh, pWidth - 2, pHeight)
    End If

    i = Sin(0.1 * counter) * (pHeightHalf - 1) + pHeightHalf

    Ret = Api_SelectObject(hDCh, hPenRed)
    Ret = Api_MoveToEx(hDCh, pWidth - 3, oldY, 0)
    Ret = Api_LineTo(hDCh, pWidth - 2, i)
    Ret = Api_SelectObject(hDCh, hPenBlack)
    Ret = Api_BitBlt(phDC, 0, 0, pWidth, pHeight, hDCh, 0, 0, SRCCOPY)

    counter = counter + 1
    oldY = i
End Sub

'=======================================
'=
'=======================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
    Var Ret As Long

    Ret = Api_ReleaseDC(Picture1.GethWnd, phDC)
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop : End
```

透明･透過のテスト(ＩＩ)

SetWindowLong ウィンドウに関するデータをセット
GetWindowLong ウィンドウに関するデータを取得
SetLayeredWindowAttributes 透明なウィンドウを作成 (Windows95/98には対応していません！)

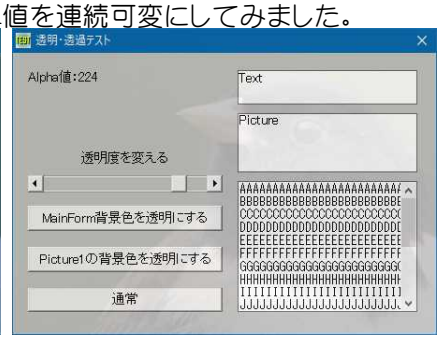
通常フォーム表示

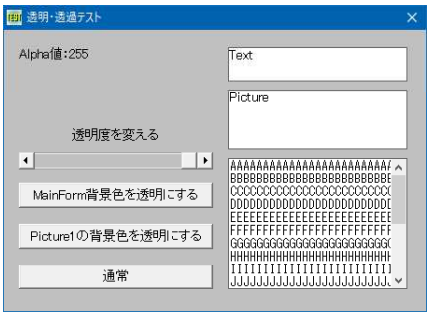
フォーム全体が透けている。透過度(アルファ値175)

フォーム(背景色:灰色)を透明に
デスクトップアイコン等が見えている

ピクチャボックス背景色(白で)を指定し透明に
TEXTBOX、PICTUREBOX、LISTBOXが透けている

BUTTON1をSCROOL1に変え、ALPHA値を連続可変にしてみました。






```
'=====================================
'=  透明･透過のテスト(ＩＩ)
'=  ※256色以下では透明なWindowは作成されない
'=  SetLayeredWindowAttributesはWindows95/98では不可
'=    (SetLayeredWinAttri.bas)
'=====================================
#include "Windows.bi"

' 指定されたウィンドウの属性を変更。また、拡張ウィンドウメモリの指定されたオフセットの32ビット値を書き換えることができる
Declare Function Api_SetWindowLong& Lib "user32" Alias "SetWindowLongA" (ByVal hWnd&,
ByVal nIndex&, ByVal dwNewLong&)

' 指定されたウィンドウに関しての情報を取得。また、拡張ウィンドウメモリから、指定されたオフセットにある32ビット値を取得することもできる
Declare Function Api_GetWindowLong& Lib "user32" Alias "GetWindowLongA" (ByVal hWnd&,
ByVal nIndex&)

' レイヤード ウィンドウの不透明および透明のカラーキーを設定
Declare Function Api_SetLayeredWindowAttributes& Lib "user32" Alias
"SetLayeredWindowAttributes" (ByVal hWnd&, ByVal crKey&, ByVal bAlpha&, ByVal dwFlags&)

#define GWL_EXSTYLE (-20)                    '拡張ウィンドウスタイル(nIndexの定数)
#define LWA_COLORKEY 1                       'crKeyを透明色として使う(dwFlagsの定数)
#define LWA_ALPHA 2                          'bAlphaをアルファー値として使う
#define WS_EX_LAYERED &H80000                '透明なウィンドウ属性(Windows2000以上)
```

```
Var shared Text(2) As Object
Var Shared Button(2) As Object
Var shared List1 As Object
Var shared Picture1 As Object
Var shared HScroll1 As Object
Picture1.Attach GetDlgItem("Picture1")
List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
HScroll1.Attach GetDlgItem("HScroll1")

For i = 0 To 2
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1)))
    Text(i).SetFontSize 14
    Button(i).Attach GetDlgItem("Button" & Trim$(Str$(i + 1)))
    Button(i).SetFontSize 14
Next

Var shared hWnd As Long
Var shared Alpha As Byte
Var shared CrLf$ As String

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    CrLf$ = Chr$(13, 10)
    Text(0).SetWindowtext ""
    Picture1.Symbol(0, 0), "Picture", 1, 1
    hWnd = GethWnd

    For i = 1 To 20
        List1.AddString String$(40, Chr$(&H40 + i))
    Next

    HScroll1.SetScrollRange 25, 255
    HScroll1.SetScrollStep 5, 1
    HScroll1.SetScrollPos 255
End Sub

'=====================================
'=  ウィンドウ全体を半透明にする(クリック毎に透過度を可変)
'=====================================
Declare Sub HScroll1_Change edecl ()
Sub HScroll1_Change()
    Var dwStyle As Long
    Var Ret As Long

    Alpha = HScroll1.GetScrollPos

    '拡張ウィンドウスタイルにWS_EX_LAYEREDを追加する
    dwStyle = Api_GetWindowLong(hWnd, GWL_EXSTYLE)
    dwStyle = dwStyle Or WS_EX_LAYERED

    Ret = Api_SetWindowLong(hWnd, GWL_EXSTYLE, dwStyle)
    Ret = Api_SetLayeredWindowAttributes(hWnd, 0, Alpha, LWA_ALPHA)

    Text(0).SetWindowtext "Alpha値:" & Trim$(Str$(Alpha))
End Sub

'=====================================
'=  MainForm背景色を透明にする
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var dwStyle As Long
    Var Ret As Long

    '拡張ウィンドウスタイルにWS_EX_LAYEREDを追加する
    dwStyle = Api_GetWindowLong(hWnd, GWL_EXSTYLE)
    dwStyle = dwStyle Or WS_EX_LAYERED
```

```
    Ret = Api_SetWindowLong(hWnd, GWL_EXSTYLE, dwStyle)

    '透明色を指定して透明にする
    Ret = Api_SetLayeredWindowAttributes(hWnd, GetBackColor, 0, LWA_COLORKEY)
    Text(0).SetWindowtext "メインフォームの背景色を取得(GetBackColor)し、その色を指定して透明に••"
End Sub

'=====================================
'=  Picture1の背景色を透明にする
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var dwStyle As Long
    Var Ret As Long

    '拡張ウィンドウスタイルにWS_EX_LAYEREDを追加する
    dwStyle = Api_GetWindowLong(hWnd, GWL_EXSTYLE)
    dwStyle = dwStyle Or WS_EX_LAYERED
    Ret = Api_SetWindowLong(hWnd, GWL_EXSTYLE, dwStyle)

    'Picture1の背景色を取得し、その色を透明にする↓
    Ret = Api_SetLayeredWindowAttributes(hWnd, Picture1.GetBackColor, 0, LWA_COLORKEY)
    Text(0).SetWindowtext "ピクチャボックスの背景色を取得(GetBackColor)し、その色を指定して透明に••"
End Sub

'=====================================
'=  ウィンドウ全体を通常に戻す
'=====================================
Declare Sub Button3_on edecl ()
Sub Button3_on()
    Var dwStyle As Long
    Var Ret As Long

    '拡張ウィンドウスタイルにWS_EX_LAYEREDを追加する
    dwStyle = API_GETWINDOWLONG&( hWnd, GWL_EXSTYLE)
    dwStyle = dwStyle Or WS_EX_LAYERED
    Ret = Api_SetWindowLong(hWnd, GWL_EXSTYLE, dwStyle)

    'ウィンドウ全体を通常に戻す                    ↓透明度(255:通常)
    Ret = Api_SetLayeredWindowAttributes(hWnd, 0, 255, LWA_ALPHA)
    Text(0).SetWindowtext "Alpha値:" & Trim$(Str$(255))
End Sub

'=====================================
'=  イベント
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

透明•透過のテスト(II)失敗の巻

SetWindowLong Windowに関するデータをセット
GetWindowLong Windowに関するデータを取得
SetLayeredWindowAttributes 透明なWindowを作成
SendMessage ウィンドウにメッセージを送信
MoveWindow 指定されたウィンドウの位置およびサイズを変更

VBのFlexGridの代用としてListBoxを下線ありで使用していますが、行間隔が窮屈です。
透過のテストをしていて、フォームを2枚重ね、下側の白いフォームに罫線を描画し、
上側のフォームに貼り付けたListBoxを白を透過色にしたらどうなるだろう？の実験•••
ListBox内のフォントサイズを14、行間隔を20に設定し適当なデータを用意して表示させてみました。
表示結果はまぁまぁ、でも動作が遅く実用にはならないようです。残念orz

Form2(背面)のフォームはダイアログフレーム、コントロール無しに設定しています。
白を透過させているのでキャプション(透明・透過の文字)を白く見えるように・・
MainFormとForm2の移動時の同期はとっていません！

```
'=======================================
'= 透明・透過のテスト(II)失敗の巻
'=    (SetLayeredWinAttri2.bas)
'=======================================
#include "Windows.bi"

'Windowに関するデータをセット
Declare Function Api_SetWindowLong& Lib "user32" Alias "SetWindowLongA" (ByVal hWnd&,
ByVal nIndex&, ByVal dwNewLong&)

'Windowに関するデータを取得
Declare Function Api_GetWindowLong& Lib "user32" Alias "GetWindowLongA" (ByVal hWnd&,
ByVal nIndex&)

'透明なWindowを作成
Declare Function Api_SetLayeredWindowAttributes& Lib "user32" Alias
"SetLayeredWindowAttributes" (ByVal hWnd&, ByVal crKey&, ByVal bAlpha&, ByVal dwFlags&)

' ウィンドウにメッセージを送信。この関数は、指定したウィンドウのウィンドウプロシージャが処理を終了するまで制御を返さない
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

' 指定されたウィンドウの位置およびサイズを変更
Declare Function Api_MoveWindow& Lib "user32" Alias "MoveWindow" (ByVal hWnd&, ByVal X&,
ByVal Y&, ByVal nWidth&, ByVal nHeight&, ByVal bRepaint&)

#define GWL_EXSTYLE (-20)                    '拡張ウィンドウスタイル(nIndexの定数)
#define LB_SETITEMHEIGHT &H1A0               'リストボックス項目の高さを設定
#define LWA_ALPHA 2                          'bAlphaをアルファー値として使う
#define LWA_COLORKEY 1                       'crKeyを透明色として使う(dwFlagsの定数)
#define WS_EX_LAYERED &H80000                '透明なウィンドウ属性(Windows2000以上)

Var shared Form2 As Object
Var shared List1 As Object

List1.Attach GetDlgItem("List1") : List1.SetFontSize 14

'=======================================
'=
'=======================================
declare sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var dwStyle As Long
    Var Ret As Long

    Form2.CreateWindow "Form2", 0

    'Form2に罫線を引く
    For v = 62 To 362 step 20
        Form2.Line(0, v)-(List1.GetWidth, v), , 14
```

```
    Next
    Form2.Line( 62, 62)-( 62, 362), , 14
    Form2.Line(192, 62)-(192, 362), , 14
    Form2.Line(258, 62)-(258, 362), , 14
    Form2.Line(488, 62)-(488, 362), , 14
    Form2.Line(648, 62)-(648, 362), , 14

    SetTopMostWindow -1

    ItemHeight = 20
    Ret = Api_SendMessage&(List1.GethWnd, LB_SETITEMHEIGHT, 0, ByVal ItemHeight)

    List1.SetWindowSize 748, 310
    List1.Resetcontent

    Count = 0
    FF = FreeFile
    Open "TEST.CSV" For Input As #FF
    While Not Eof(#FF)
        Count = Count + 1
        ZD1$ = Format$(Count,"#####") & " "
        Input #FF, ZD2$ : ZD2$ = Left$(Kacnv$(ZD2$) & Space$(18), 18) & " "
        Input #FF, ZD3$ : ZD3$ = Left$(ZD3$, 8) & " "
        Input #FF, ZD4$ : ZD4$ = Left$(Kacnv$(ZD4$) & Space$(32), 32) & " "
            If Asc(Mid$(ZD4$, 32, 1)) > &H81 Then    ZD4$ = Left$(ZD4$, 31) & "  "
        Input #FF, ZD5$ : ZD5$ = Left$(Kacnv$(ZD5$) & Space$(22), 22) & " "
            If Asc(Mid$(ZD5$, 22, 1)) > &H81 Then    ZD5$ = Left$(ZD5$, 21) & "  "
        Input #FF, ZD6$ : ZD6$ = Left$(ZD6$ & Space$(13), 13) & " "
        List1.AddString ZD1$ & ZD2$ & ZD3$ & ZD4$ & ZD5$ & ZD6$
    Wend

    '拡張ウィンドウスタイルにWS_EX_LAYEREDを追加する
    dwStyle = Api_GetWindowLong&(GethWnd, GWL_EXSTYLE)
    dwStyle = dwStyle Or WS_EX_LAYERED
    Ret = Api_SetWindowLong&(GethWnd, GWL_EXSTYLE, dwStyle)

    '透明色を指定して透明にする
    Ret = Api_SetLayeredWindowAttributes&(GethWnd, Form2.GetBackColor, 0, LWA_COLORKEY)
    Form2.ShowWindow -1
    ShowWindow -1
End Sub

'=====================================
'=  イベント
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

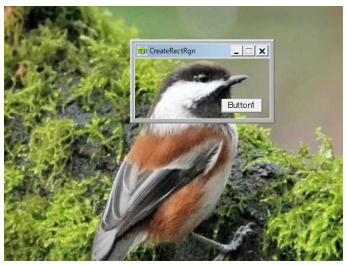
「TEST.CSV」は、下記のとおり

```
相川　欽助,087-0021,根室市幸町1丁目3－1,,
安達　邦明,003-0022,札幌市白石区南郷通14丁目2－1,山田マンション302,011-123-4567
井川　国安,090-0833,北見市とん田,,
江藤　五朗,002-8026,札幌市北区篠路6条8丁目,,
大山　大寒,061-2273,札幌市南区豊滝234,,
鎌田　浩介,080-0038,帯広市西8条北3丁目,,
亀田　勇作,040-0001,函館市五稜郭,,
川谷　光男,061-2285,札幌市南区藤野5条4丁目1－2,,
川中　美由紀,005-0841,札幌市南区石山1条6丁目,,
北村　末治,003-0022,札幌市白石区南郷通13丁目南5－1,山並パレス205,011-863-1234
木村　謙吉,093-0021,網走市南十一条西1丁目,,
近田　祐介,006-0811,札幌市手稲区前田1条8丁目,,
斉藤　花子,002-8033,札幌市北区西茨戸3条8丁目,,
佐々木　小次郎,003-0021,札幌市白石区栄通2丁目,宮本ビル1F,
佐々木　守靖,077-0000,留萌市,,
佐野　元義,003-0022,札幌市白石区南郷通11丁目南4,,
```

島田　古助,061-2274,札幌市南区小金湯,,
鈴木　敬一,099-3602,斜里郡小清水町東野,,
田中　各男,003-0021,札幌市白石区栄通1丁目,,
津田　健一,003-0021,札幌市白石区栄通15丁目4－1,,
所　常蔵,097-0006,稚内市新港1丁目,,
富並　雄一,003-0028,札幌市白石区平和通(南),,
内藤　国の助,003-0022,札幌市白石区南郷通11丁目南8－1,アトムマンション102,011-861-4567
畑中　俊輔,004-0053,札幌市厚別区厚別中央3丁目,,011-883-6587
宮本　武蔵,062-0932,札幌市豊平区平岸2条9丁目,,
森田　杉作,093-0017,網走市南7条西1丁目,,
安田　聖子,040-0001,函館市五稜郭町123,,
ラリアット株式会社,006-0836,札幌市手稲区曙6条3丁目21－1,,
和光　留夫,097-0021,稚内市港1－6－5,,
渡辺　昭夫,003-0021,札幌市白石区栄通,,

透明なフォーム

CreateRectRgn 長方形のリージョンを作成
CombineRgn 既存の二つの領域を結合して新しい領域を作成
SetWindowRgn 指定の領域をウィンドウ領域として設定
GetSystemMetrics さまざまなシステムメトリックの値とシステムの現在の構成を取得



例では、フォームに1個のButtonを貼り付けフォームの内側を透明にしています。

```
'=====================================
'= 透明なフォーム
'=    (CreateRectRgn2.bas)
'=====================================
#include "Windows.bi"

' 長方形のリージョンを作成
Declare Function Api_CreateRectRgn& Lib "gdi32" Alias "CreateRectRgn" (ByVal nLeftRect&,
ByVal nTopRect&, ByVal nRightRect&, ByVal nBottomRect&)

' 既存の二つの領域を結合して新しい領域を作成
Declare Function Api_CombineRgn& Lib "gdi32" Alias "CombineRgn" (ByVal hRgnDest&, ByVal
hRgnSrc1&, ByVal hRgnSrc2&, ByVal nCombineMode&)

' 指定の領域をウィンドウ領域として設定
Declare Function Api_SetWindowRgn& Lib "user32" Alias "SetWindowRgn" (ByVal hWnd&, ByVal
hRgn&, ByVal bRedraw&)

' さまざまなシステムメトリックの値とシステムの現在の構成を取得
Declare Function Api_GetSystemMetrics& Lib "user32" Alias "GetSystemMetrics" (ByVal
nIndex&)

#define RGN_DIFF 4                          'HRGNSRC1からHRGNSRC2を除いた領域
#define RGN_OR 2                            'リージョン同士のOR結合
#define SM_CXFRAME 32                       'サイズ可変ウィンドウの境界線のX方向の幅
#define SM_CYFRAME 33                       'サイズ可変ウィンドウの境界線のY方向の幅
#define SM_CYBORDER 6                       'サイズ固定ウィンドウの境界線のY方向の幅
#define SM_CYCAPTION 4                      'キャプションの高さ

Var Shared Button1 As Object
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
```

```
Declare Sub TransparentForm ()
Sub TransparentForm()
    Var outer_rgn As Long
    Var inner_rgn As Long
    Var wid As Single
    Var hgt As Single
    Var border_width As Single
    Var title_height As Single
    Var combined_rgn As Long
    Var ctl_left As Single
    Var ctl_top As Single
    Var ctl_right As Single
    Var ctl_bottom As Single
    Var control_rgn As Long
    Var Ret As Long

    'フォームの外側
    wid = GetWidth
    hgt = GetHeight
    outer_rgn = Api_CreateRectRgn(0, 0, wid, hgt)

    'フォームの内側
    border_width = Api_GetSystemMetrics(SM_CXFRAME)
    title_height = Api_GetSystemMetrics(SM_CYFRAME) + Api_GetSystemMetrics(SM_CYBORDER)
+ Api_GetSystemMetrics(SM_CYCAPTION)
    inner_rgn = Api_CreateRectRgn(border_width, title_height, wid - border_width, hgt -
border_width)

    '外側から内側の領域を引き算
    combined_rgn = Api_CreateRectRgn(0, 0, 0, 0)
    Ret = Api_CombineRgn(combined_rgn, outer_rgn, inner_rgn, RGN_DIFF)

    ctl_left = border_width + 144
    ctl_top = title_height + 66
    ctl_right = ctl_left + 68
    ctl_bottom = ctl_top + 24

    control_rgn = Api_CreateRectRgn(ctl_left, ctl_top, ctl_right, ctl_bottom)
    Ret = Api_CombineRgn(combined_rgn, combined_rgn, control_rgn, RGN_OR)

    'ウィンドウを指定領域に制限
    Ret = Api_SetWindowRgn(GethWnd, combined_rgn, True)
End Sub

'=======================================
'=
'=======================================
Declare Sub MainForm_Resize edecl ()
Sub MainForm_Resize()
    If GetWidth < 240 Or GetHeight < 140 Then SetWindowSize 240, 140

    TransparentForm
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

ドキュメント名の設定

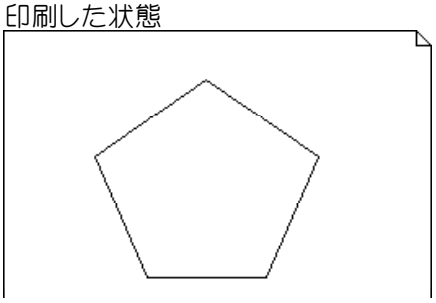
CreateDC 指定されたデバイスのデバイスコンテキストを、指定された名前で作成
DeleteDC 指定されたデバイスコンテキストを削除

Polygon 多角形の描画
StartDoc 印刷ジョブを開始
EndDoc 印刷ジョブを終了
StartPage プリンタドライバがデータを受け取る準備をさせる
EndPage 1ページ書き込みの終了を通知
GetDC ディスプレイデバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放

起動時

設定したドキュメント名が表示される

印刷した状態



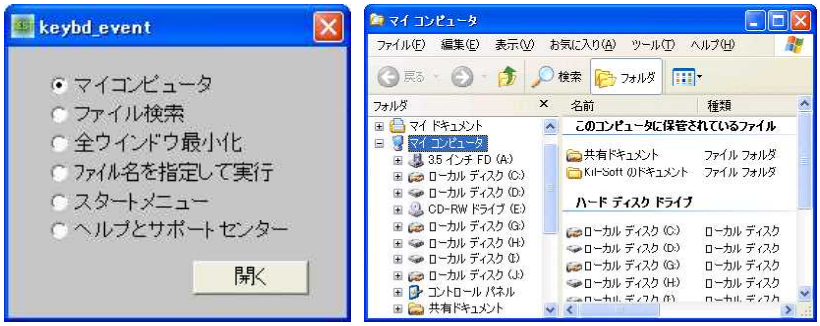
```
'=====================================
'=ドキュメント名の設定
'=    (DocName.bas)
'=====================================
#include "Windows.bi"
Type POINTAPI
    x   As Long
    y   As Long
End Type

Type DOCINFO
    cbSize              As Long
    lpszDocName         As Long
    lpszOutput          As Long
    lpszDatatype        As Long
    fwType              As Long
End Type

' 指定されたデバイスのデバイスコンテキストを、指定された名前で作成
Declare Function Api_CreateDC& Lib "gdi32" Alias "CreateDCA" (ByVal lpDriverName$,
lpDeviceName As Any, lpOutput As Any, ByVal lpInitData As Any)

' 指定されたデバイスコンテキストを削除
Declare Function Api_DeleteDC& Lib "gdi32" Alias "DeleteDC" (ByVal hDC&)

' 直線により接続された2つ以上の頂点から成っているポリゴンを引く
Declare Function Api_Polygon& Lib "gdi32" Alias "Polygon" (ByVal hDC&, lpPoint As Any,
ByVal nCount&)

' 印刷ジョブを開始
Declare Function Api_StartDoc& Lib "gdi32" Alias "StartDocA" (ByVal hDC&, lpdi As
DOCINFO)

' プリンタドライバがデータを受け取る準備をさせる
Declare Function Api_StartPage& Lib "gdi32" Alias "StartPage" (ByVal hDC&)

' 1ページ書き込みの終了を通知
Declare Function Api_EndPage& Lib "gdi32" Alias "EndPage" (ByVal hDC&)

' 印刷ジョブを終了
Declare Function Api_EndDoc& Lib "gdi32" Alias "EndDoc" (ByVal hDC&)

' 指定されたウィンドウのデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

Var Shared Button1 As Object
```

```
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var Shared pa(4) As POINTAPI
Var Shared n As Integer

'=====================================
'=
'=====================================
Declare Sub Polygon_Set()
Sub Polygon_Set()
    pa(0).x = 110 * n
    pa(0).y = 20 * n
    pa(1).x = 45 * n
    pa(1).y = 65 * n
    pa(2).x = 75 * n
    pa(2).y = 135 * n
    pa(3).x = 145 * n
    pa(3).y = 135 * n
    pa(4).x = 175 * n
    pa(4).y = 65 * n
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var hDC As Long
    Var Ret As Long

    hDC = Api_GetDC(GethWnd)

    n = 1
    Polygon_Set
    Ret = Api_Polygon(hDC, pa(0), 5)

    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var PrinterDC As Long
    Var di As DOCINFO
    Var Ret As Long

    di.cbSize = Len(di)
    di.lpszDocName = StrAdr("API-Guide Code Demonstration" & Chr$(0))
    di.lpszOutput = StrAdr(Chr$(0))
    di.lpszDatatype = StrAdr(Chr$(0))

    PrinterDC = Api_CreateDC(ByVal 0, "pdfFactory Pro", Chr$(0), ByVal 0)

    Ret = Api_StartDoc(PrinterDC, di)
    Ret = Api_StartPage(PrinterDC)

    n = 10
    Polygon_Set
    Ret = Api_Polygon(PrinterDC, pa(0), 5)

    Ret = Api_EndPage(PrinterDC)
    Ret = Api_EndDoc(PrinterDC)

    Ret = Api_DeleteDC(hDC)
End Sub
```

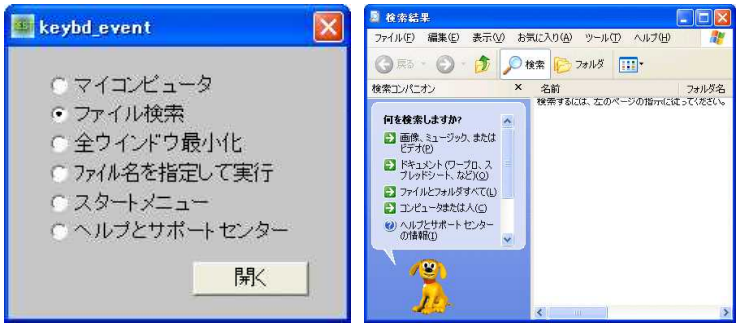
```
'====================================
'=
'====================================
While 1
    WaitEvent
Wend
Stop
End
```

特殊キーの状態を設定

keybd_eventでキーストロークをシミュレートします。
keybd_event キーストロークを合成する関数



ラジオボタンをチェックし開くをクリックした状態を右図に示します。全ウインドウ最小化の状態は省略しています。



スタートメニューは分けて表示しています。

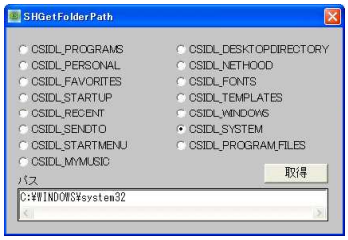

```
'=======================================
'= 特殊キーの状態を設定
'=    (keybd_event.bas)
'=======================================
#include "Windows.bi"

' キーストロークを合成する関数の宣言
Declare Sub Api_keybd_event Lib "user32" Alias "keybd_event" (ByVal bVk As Byte, ByVal
bScan As Byte, ByVal dwFlags&, ByVal dwExtraInfo&)

#define VK_LWIN &H5B                        'Windowsキー(左)
#define VK_RWIN &H5C                        'Windowsキー(右)
#define VK_APPS &H5D                        'アプリケーションキー
#define KEYEVENTF_KEYUP &H2                 'キーを放す

Var Shared Radio(5) As Object
Var Shared Button1 As Object

For i = 0 To 5
    Radio(i).Attach GetDlgItem("Radio" & Trim$(Str$(i + 1)))
    Radio(i).SetFontSize 14
Next
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=======================================
'=
'=======================================
Declare Function Index bdecl () As Integer
Function Index()
    Index = Val(Mid$(GetDlgRadioSelect("Radio1"), 6)) - 1
End Function

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var VK_ACTION As Long

    Select Case Index
        Case 0
            VK_ACTION = &H45             'E
        Case 1
            VK_ACTION = &H46             'F
        Case 2
            VK_ACTION = &H4D             'M
        Case 3
            VK_ACTION = &H52             'R
        Case 4
            VK_ACTION = &H5B             'Window(Start Menu Button)
        Case 5
            VK_ACTION = &H70             'F1(ascii 112)
    End Select

    Api_keybd_event VK_LWIN, 0, 0, 0     'Window
    Api_keybd_event VK_ACTION, 0, 0, 0
    Api_keybd_event VK_LWIN, 0, KEYEVENTF_KEYUP, 0
```

```
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

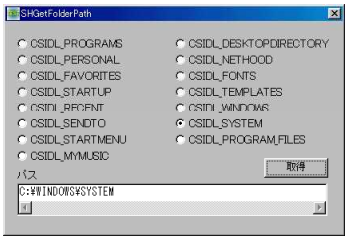
特殊フォルダパスの取得(Ⅰ)

特殊なフォルダパスを取得します。
SHGetFolderPath CSIDLからパスを取得

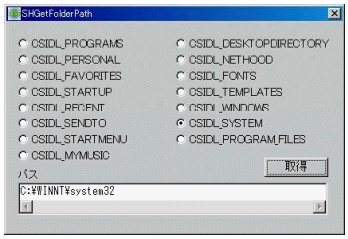
WindowsXP



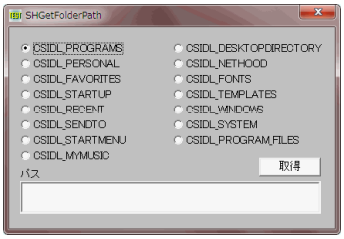
Windows98



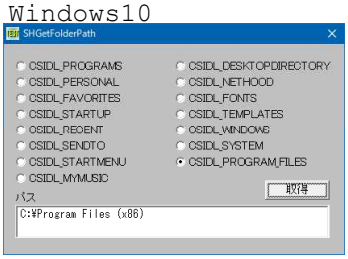
Windows2000



Windows 7



Windows10




```
'=====================================
'=  特殊フォルダのパス名等の取得
'=    (SHGetFolderPath.bas)
'=====================================
#include "Windows.bi"

' CSIDLからパスを取得する関数の宣言
Declare Function Api_SHGetFolderPath& Lib "ShFolder" Alias "SHGetFolderPathA" (ByVal
hwndOwner&, ByVal nFolder&, ByVal hToken&, ByVal dwFlags&, ByVal pszPath$)

' 取り出すパスの種類を指定する定数
#define SHGFP_TYPE_CURRENT 0                    '取り出すパスの種類を指定する定数
#define SHGFP_TYPE_DEFAULT 1                    '取り出すパスの種類を指定する定数

' CSIDLを示す定数の宣言
#define CSIDL_INTERNET &H1                      'InternetExplorer(仮想フォルダ)
#define CSIDL_PROGRAMS &H2                      'プログラム(ファイルシステムディレクトリ)
#define CSIDL_CONTROLS &H3                      'コントロールパネル(仮想フォルダ)
#define CSIDL_PRINTERS &H4                      'プリンタ(仮想フォルダ)
#define CSIDL_PERSONAL &H5                      'マイドキュメント(ファイルシステムディレクトリ)
#define CSIDL_FAVORITES &H6                     'お気に入り(ファイルシステムディレクトリ)
#define CSIDL_STARTUP &H7                       'スタートアップ(ファイルシステムディレクトリ)
#define CSIDL_RECENT &H8                        '最近使ったファイル(ファイルシステムディレクトリ)
#define CSIDL_SENDTO &H9                        'SEndTo(ファイルシステムディレクトリ)
#define CSIDL_BITBUCKET &HA                     'ゴミ箱(仮想フォルダ)
#define CSIDL_STARTMENU &HB                     'スタートメニュー(ファイルシステムディレクトリ)
#define CSIDL_MYMUSIC &HD                       'マイミュージック(ファイルシステムディレクトリ)
#define CSIDL_DESKTOPDIRECTORY &H10             'デスクトップ上のファイルオブジェクトを格納するフォルダ
                                                (ファイルシステムディレクトリ)
#define CSIDL_DRIVES &H11                       'マイコンピュータ(仮想フォルダ)
#define CSIDL_NETWORK &H12                      'ネットワークコンピュータ(仮想フォルダ)
```

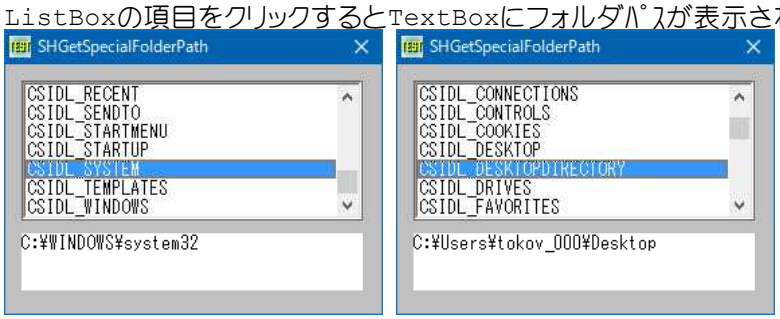
```
#define CSIDL_NETHOOD &H13                  'NetHood(ファイルシステムディレクトリ)
#define CSIDL_FONTS &H14                    'Fonts(フォントを含む仮想フォルダ)
#define CSIDL_TEMPLATES &H15                'ドキュメントテンプレートが格納されるフォルダ(ファイルシ
                                            ステムディレクトリ)
#define CSIDL_WINDOWS &H24                  'Windowsディレクトリ
#define CSIDL_SYSTEM &H25                   'Windows Systemディレクトリ
#define CSIDL_PROGRAM_FILES &H26            'Program Filesフォルダ

Var Shared Edit1 As Object
Var Shared Text1 As Object
Var Shared Button1 As Object
Var Shared Radio(14) As Object

'=====================================
'=
'=====================================
Declare Sub Mainform_Start edecl ()
Sub Mainform_Start()
    Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
    Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
    Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

    For i = 0 To 14
        Radio(i).Attach GetDlgItem("Radio" & Trim$(Str$(i + 1)))
        Radio(i).SetFontSize 14
    Next i
    ShowWindow -1
End Sub

'=====================================
'=
'=====================================
Declare Function Index bdecl () As Integer
Function Index()
    Index = val(Mid$(GetDlgRadioSelect("Radio1"), 6)) -1
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var fPath As String * 260
    Var Str As String
    Var CSIDL As Long
    Var Ret As Long

    Select Case Index
        Case 0                              'CSIDL_PROGRAMS
            CSIDL = &H2
        Case 1                              'CSIDL_PERSONAL
            CSIDL = &H5
        Case 2                              'CSIDL_FAVORITES
            CSIDL = &H6
        Case 3                              'CSIDL_STARTUP
            CSIDL = &H7
        Case 4                              'CSIDL_RECENT
            CSIDL = &H8
        Case 5                              'CSIDL_SENDTO
            CSIDL = &H9
        Case 6                              'CSIDL_STARTMENU
            CSIDL = &HB
        Case 7                              'CSIDL_MYMUSIC
            CSIDL = &HD
        Case 8                              'CSIDL_DESKTOPDIRECTORY
            CSIDL = &H10
        Case 9                              'CSIDL_NETHOOD
            CSIDL = &H13
        Case 10                             'CSIDL_FONTS
            CSIDL = &H14
```

```
        Case 11                                 'CSIDL_TEMPLATES
            CSIDL = &H15
        Case 12                                 'CSIDL_WINDOWS
            CSIDL = &H24
        Case 13                                 'CSIDL_SYSTEM
            CSIDL = &H25
        Case 14                                 'CSIDL_PROGRAM_FILES
            CSIDL = &H26
    End Select

    ' フォルダのパスを取得
    Ret = Api_SHGetFolderPath(0, CSIDL, SHGFP_TYPE_CURRENT, 0, fPath)

    ' フォルダのパスを表示
    Edit1.SetWindowText Left$(fPath, InStr(fPath, Chr$(0)) - 1)
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

特殊フォルダパスの取得(II)

特殊フォルダパスを取得します。
SHGetSpecialFolderPath 特殊フォルダのパスを取得

ListBoxの項目をクリックするとTextBoxにフォルダパスが表示されます。




```
'=====================================
'= 特殊フォルダのパス名等の取得
'=    (SHGetSpecialFolderPath.bas)
'=====================================
#include "Windows.bi"

' 特殊フォルダのパスを取得
Declare Function Api_SHGetSpecialFolderPath& Lib "shell32" Alias
"SHGetSpecialFolderPathA" (ByVal hwndOwner&, ByVal lpszPath$, ByVal nFolder&, ByVal
fCreate&)

#define CSIDL_ADMINTOOLS &H30               '管理ツール
#define CSIDL_ALTSTARTUP &H1D               '非ローカライズスタートアップ
#define CSIDL_APPDATA &H1A                  'アプリケーションデータ
#define CSIDL_BITBUCKET &HA                 'ゴミ箱(仮想フォルダ)
#define CSIDL_CONNECTIONS &H31              'ネットワーク接続
#define CSIDL_CONTROLS &H3                  'コントロールパネル(仮想フォルダ)
#define CSIDL_COOKIES &H21                  'クッキー(IE)
#define CSIDL_DESKTOP &H0                   'デスクトップ
#define CSIDL_DESKTOPDIRECTORY &H10         'デスクトップ上のファイルオブジェクトを格納するフォルダ
                                            (ファイルシステムディレクトリ)
#define CSIDL_DRIVES &H11                   'マイコンピュータ(仮想フォルダ)
#define CSIDL_FAVORITES &H6                 'お気に入り(ファイルシステムディレクトリ)
#define CSIDL_FONTS &H14                    'Fonts(フォントを含む仮想フォルダ)
```

```
#define CSIDL_HISTORY &H22                  '履歴(IE)
#define CSIDL_INTERNET &H1                  'InternetExplorer(仮想フォルダ)
#define CSIDL_INTERNET_CACHE &H20           'インターネットキャッシュ
#define CSIDL_LOCAL_APPDATA &H1C            'ローカルアプリケーションデータ
#define CSIDL_MYMUSIC &HD                   'マイミュージック(ファイルシステムディレクトリ)
#define CSIDL_MYPICTURES &H27               'マイピクチャ
#define CSIDL_MYVIDEO &HE                   'マイビデオ
#define CSIDL_NETHOOD &H13                  'NetHood(ファイルシステムディレクトリ)
#define CSIDL_NETWORK &H12                  'ネットワークコンピュータ(仮想フォルダ)
#define CSIDL_PERSONAL &H5                  'マイドキュメント(ファイルシステムディレクトリ)
#define CSIDL_PRINTERS &H4                  'プリンタ(仮想フォルダ)
#define CSIDL_PRINTHOOD &H1B                'プリンタ
#define CSIDL_PROFILE &H28                  'プロファイル
#define CSIDL_PROGRAM_FILES &H26            'ProgramFilesフォルダ
#define CSIDL_PROGRAMS &H2                  'プログラム(ファイルシステムディレクトリ)
#define CSIDL_RECENT &H8                    '最近使ったファイル(ファイルシステムディレクトリ)
#define CSIDL_SENDTO &H9                    'SEndTo(ファイルシステムディレクトリ)
#define CSIDL_STARTMENU &HB                 'スタートメニュー(ファイルシステムディレクトリ)
#define CSIDL_STARTUP &H7                   'スタートアップ(ファイルシステムディレクトリ)
#define CSIDL_SYSTEM &H25                   'WindowsSystemディレクトリ
#define CSIDL_TEMPLATES &H15                'ドキュメントテンプレートが格納されるフォルダ(ファイルシ
                                            ステムディレクトリ)
#define CSIDL_WINDOWS &H24                  'Windowsディレクトリ
#define MAX_PATH 260

Var Shared List1 As Object
Var Shared Text1 As Object
List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14

'====================================
'=
'====================================
Declare Sub List1_Click edecl ()
Sub List1_Click()
    Var FolderNo As Integer
    Var Buff As String
    Var lFolder As Long
    Var Ret As Long

    Text1.SetWindowText ""
    FolderNo = List1.GetCursel
    Select Case FolderNo
        Case 0
            lFolder = CSIDL_ADMINTOOLS
        Case 1
            lFolder = CSIDL_ALTSTARTUP
        Case 2
            lFolder = CSIDL_APPDATA
        Case 3
            lFolder = CSIDL_BITBUCKET
        Case 4
            lFolder = CSIDL_CONNECTIONS
        Case 5
            lFolder = CSIDL_CONTROLS
        Case 6
            lFolder = CSIDL_COOKIES
        Case 7
            lFolder = CSIDL_DESKTOP
        Case 8
            lFolder = CSIDL_DESKTOPDIRECTORY
        Case 9
            lFolder = CSIDL_DRIVES
        Case 10
            lFolder = CSIDL_FAVORITES
        Case 11
            lFolder = CSIDL_FONTS
        Case 12
            lFolder = CSIDL_HISTORY
        Case 13
```

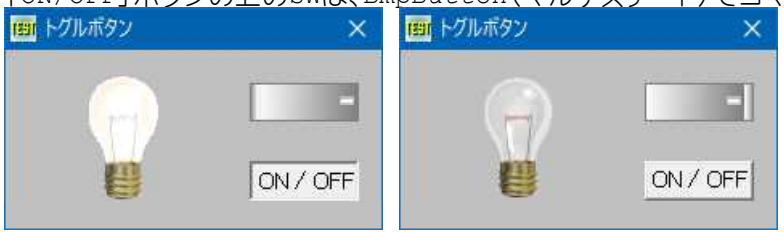
```
            lFolder = CSIDL_INTERNET
        Case 14
            lFolder = CSIDL_INTERNET_CACHE
        Case 15
            lFolder = CSIDL_LOCAL_APPDATA
        Case 16
            lFolder = CSIDL_MYMUSIC
        Case 17
            lFolder = CSIDL_MYPICTURES
        Case 18
            lFolder = CSIDL_MYVIDEO
        Case 19
            lFolder = CSIDL_NETHOOD
        Case 20
            lFolder = CSIDL_NETWORK
        Case 21
            lFolder = CSIDL_PERSONAL
        Case 22
            lFolder = CSIDL_PRINTERS
        Case 23
            lFolder = CSIDL_PRINTHOOD
        Case 24
            lFolder = CSIDL_PROFILE
        Case 25
            lFolder = CSIDL_PROGRAM_FILES
        Case 26
            lFolder = CSIDL_PROGRAMS
        Case 27
            lFolder = CSIDL_RECENT
        Case 28
            lFolder = CSIDL_SENDTO
        Case 29
            lFolder = CSIDL_STARTMENU
        Case 30
            lFolder = CSIDL_STARTUP
        Case 31
            lFolder = CSIDL_SYSTEM
        Case 32
            lFolder = CSIDL_TEMPLATES
        Case 33
            lFolder = CSIDL_WINDOWS
    End Select

    Buff = String$(MAX_PATH, Chr$(0))
    Ret = Api_SHGetSpecialFolderPath(0, Buff, lFolder, 0)

    Text1.SetWindowText Left$(Buff, InStr(Buff, Chr$(0)) - 1)
End Sub

'====================================
'=
'====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    List1.AddString "CSIDL_ADMINTOOLS"
    List1.AddString "CSIDL_ALTSTARTUP"
    List1.AddString "CSIDL_APPDATA"
    List1.AddString "CSIDL_BITBUCKET"
    List1.AddString "CSIDL_CONNECTIONS"
    List1.AddString "CSIDL_CONTROLS"
    List1.AddString "CSIDL_COOKIES"
    List1.AddString "CSIDL_DESKTOP"
    List1.AddString "CSIDL_DESKTOPDIRECTORY"
    List1.AddString "CSIDL_DRIVES"
    List1.AddString "CSIDL_FAVORITES"
    List1.AddString "CSIDL_FONTS"
    List1.AddString "CSIDL_HISTORY"
    List1.AddString "CSIDL_INTERNET"
    List1.AddString "CSIDL_INTERNET_CACHE"
    List1.AddString "CSIDL_LOCAL_APPDATA"
```

```
    List1.AddString "CSIDL_MYMUSIC"
    List1.AddString "CSIDL_MYPICTURES"
    List1.AddString "CSIDL_MYVIDEO"
    List1.AddString "CSIDL_NETHOOD"
    List1.AddString "CSIDL_NETWORK"
    List1.AddString "CSIDL_PERSONAL"
    List1.AddString "CSIDL_PRINTERS"
    List1.AddString "CSIDL_PRINTHOOD"
    List1.AddString "CSIDL_PROFILE"
    List1.AddString "CSIDL_PROGRAM_FILES"
    List1.AddString "CSIDL_PROGRAMS"
    List1.AddString "CSIDL_RECENT"
    List1.AddString "CSIDL_SENDTO"
    List1.AddString "CSIDL_STARTMENU"
    List1.AddString "CSIDL_STARTUP"
    List1.AddString "CSIDL_SYSTEM"
    List1.AddString "CSIDL_TEMPLATES"
    List1.AddString "CSIDL_WINDOWS"
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

トグルボタン

Buttonをトグルボタンにします。
SendMessage ウィンドウにメッセージを送信
BM_SETSTATE(&HF3) ボタンの反転表示状態を設定する

通常Buttonをクリックするとすぐ元に戻りますが、次にクリックされるまでその状態を保ちます。ただそれだけ••(^^;
「ON/OFF」ボタンの上のSWは、BmpButton(マルチステート)でコマンドボタンと連動させています。(お遊び••)




```
'=====================================
'=トグルボタン
'=    (ToggleButton.bas)
'=====================================
#include "Windows.bi"

' ウィンドウにメッセージを送信。この関数は、指定したウィンドウのウィンドウプロシージャが処理を終了するまで制御を返さない
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal wMsg&, ByVal wParam&, ByVal lParam&)

#define BM_SETSTATE &HF3                    'ボタンの反転表示状態を設定する

Var Shared Button1 As Object
Var Shared BmpButton1 As Object
Var Shared Picture1 As Object
Var Shared Picture2 As Object
Var Shared Bitmap As Object
BitmapObject Bitmap
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
BmpButton1.Attach GetDlgItem("BmpButton1")
```

```
Picture1.Attach GetDlgItem("Picture1")
Picture2.Attach GetDlgItem("Picture2")

Var Shared ButtonDown As Integer

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Bitmap.LoadFile "LampOff.bmp"
    Picture1.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject
    Bitmap.LoadFile "LampOn.bmp"
    Picture2.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject
End Sub

'=====================================
'=
'=====================================
Declare Sub Lamp_on ()
Sub Lamp_on()
    Picture1.ShowWindow 0
    Picture2.ShowWindow -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Lamp_off ()
Sub Lamp_off()

    Picture2.ShowWindow 0
    Picture1.ShowWindow -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    ButtonDown = not ButtonDown
    Ret = Api_SendMessage(Button1.GethWnd, BM_SETSTATE, int(ButtonDown), 0)

    If ButtonDown Then
        Lamp_on
        BmpButton1.SetState 1
    Else
        Lamp_off
        BmpButton1.SetState 0
    End If
End Sub

'=====================================
'=
'=====================================
Declare Sub BmpButton1_on edecl ()
Sub BmpButton1_on()
    If BmpButton1.GetState = 0 Then
        Lamp_off
    Else
        Lamp_on
    End If
    Button1_on
End Sub
```

```
'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

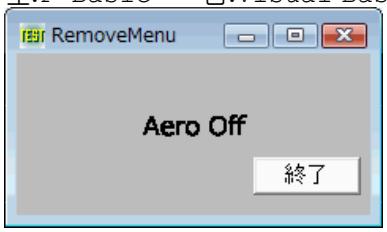
閉じる「×」ボタンの無効化（Ⅰ）

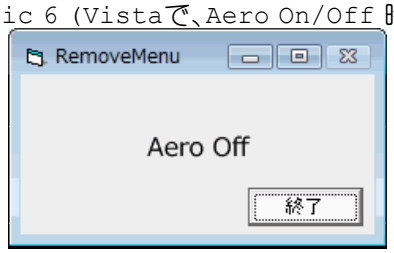
フォームの閉じる「×」ボタンを無効化します。
RemoveMenu システムメニューの削除
GetSystemMenu システムメニューのハンドル取得

起動時の状態（×が無効化）

システムメニューの初期状態　「閉じる」は削除されている

左:F-Basic　右:Visual Basic 6（Vistaで、Aero On/Off 時の「X」ボタンの状態に注目??）






```
'=====================================
'= 閉じるボタン「×」の無効化
'=    (RemoveMenu1.bas)
'=====================================
#include "Windows.bi"

#define MF_BYPOSITION &H400                    'nPositionはメニュー項目のインデックス

' システムメニューの削除
Declare Function Api_RemoveMenu& Lib "user32" Alias "RemoveMenu" (ByVal hMenu&, ByVal
nPosition&, ByVal wFlags&)

' システムメニューのハンドル取得
Declare Function Api_GetSystemMenu& Lib "user32" Alias "GetSystemMenu" (ByVal hWnd&,
ByVal bRevert&)

Var Shared Button1 As Object

Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
```

```
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var hSysMenu As Long
    Var Ret As Long

    hSysMenu = Api_GetSystemMenu(GethWnd, 0)
    Ret = Api_RemoveMenu(hSysMenu, 6, MF_BYPOSITION)
    Ret = Api_RemoveMenu(hSysMenu, 5, MF_BYPOSITION)
End Sub

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    End
End Sub

'=======================================
'=
'=======================================
Declare Sub MainForm_Resize edecl ()
Sub MainForm_Resize()
    Button1.MoveWindow GetWidth - (Button1.GetWidth + 22), GetHeight - (Button1.GetHeight
+ 50)
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

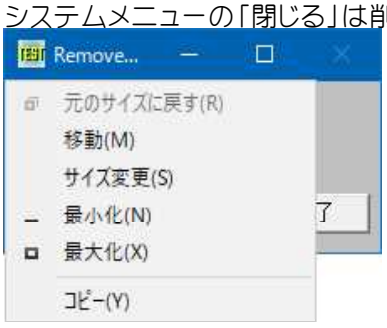
閉じる「×」ボタンの無効化（II）

フォームの閉じる「×」ボタンを無効化します。
RemoveMenu システムメニューの削除
GetSystemMenu システムメニューのハンドル取得

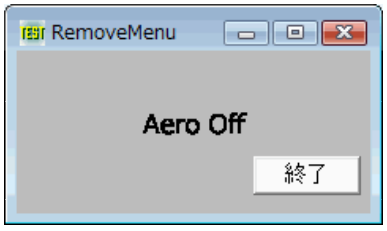
システムメニューの初期状態

起動時の状態（×が無効化）

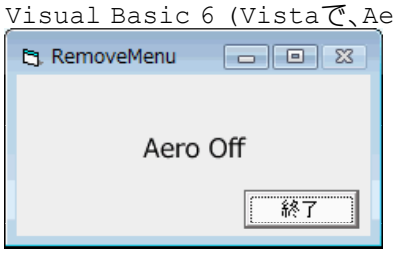
システムメニューの「閉じる」は削除されている



F-Basic



Visual Basic 6（Vistaで、Aero On/Off 時の「X」ボタンの状態に注目??）



```
'=====================================
'= 閉じる「×」ボタンの無効化（II）
'=    (RemoveMenu2.bas)
'=====================================
#include "Windows.bi"

' システムメニューのハンドル取得
Declare Function Api_GetSystemMenu& Lib "user32" Alias "GetSystemMenu" (ByVal hWnd&,
ByVal bRevert&)

' システムメニューの削除
Declare Function Api_RemoveMenu& Lib "user32" Alias "RemoveMenu" (ByVal hMenu&, ByVal
nPosition&, ByVal wFlags&)

#define SC_CLOSE &HF060                    '閉じる
#define MF_BYCOMMAND &H0                   'nPositionはメニュー項目のID

Var Shared Button1 As Object

Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var hSysMenu As Long
    Var Ret As Long

    'システムメニューハンドル取得
    hSysMenu = Api_GetSystemMenu(GethWnd, 0)

    'クローズ[×]削除
    Ret = Api_RemoveMenu(hSysMenu, SC_CLOSE, MF_BYCOMMAND)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    End
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Resize edecl ()
Sub MainForm_Resize()
    Button1.MoveWindow GetWidth - (Button1.GetWidth + 22), GetHeight - (Button1.GetHeight
+ 50)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

閉じる「×」ボタンの無効化（III）

閉じる「×」ボタンを無効化します。
GetSystemMenu システムメニューのハンドル取得
EnableMenuItem メニュー項目の有効化・無効化

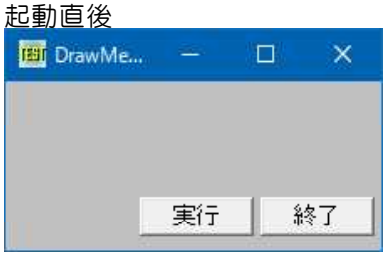
RemoveMenu は、システムメニューから項目を削除しますが、EnableMenuItem は無効化するのみ
左:起動時　　中:「× 閉じる(C)」がグレー表示　　右:MF_GRAYEDを外した状態「× 閉じる(C)」は効かない


```
'=====================================
'= 閉じる「×」ボタンの無効化(III)
'=    (EnableMenuItem.bas)
'=====================================
#include "Windows.bi"

#define MF_BYCOMMAND &H0                    'nPositionはメニュー項目のID
#define MF_BYPOSITION &H400                 'nPositionはメニュー項目のインデックス
#define MF_CHECKED &H8                      'メニュー項目にチェックをつける
#define MF_DISABLED &H2                     'アイテムを無効化(灰色表示にはならない)
#define MF_GRAYED &H1                       'グレー表示されて選択できない
#define MF_SEPARATOR &H800                  'メニュー項目はセパレータ
#define MF_STRING &H0                       '文字列
#define SC_CLOSE &HF060                     '閉じる
#define SC_MAXIMIZE &HF030                  '最大化
#define SC_MINIMIZE &HF020                  '最小化
#define SC_MOVE &HF010                      '移動
#define SC_RESTORE &HF120                   '元のサイズに戻す
#define SC_SCREENSAVE &HF140                'スクリーンセーバーを実行するメッセージ
#define SC_SIZE &HF000                      'サイズ変更

' システムメニューのハンドル取得
Declare Function Api_GetSystemMenu& Lib "user32" Alias "GetSystemMenu" (ByVal hWnd&,
ByVal bRevert&)

' メニューの項目を有効化・無効化
Declare Function Api_EnableMenuItem& Lib "user32" Alias "EnableMenuItem" (ByVal hMenu&,
ByVal wIDEnableItem&, ByVal wEnable&)

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var Ret As Long

    Ret = Api_GetSystemMenu(GethWnd, 0)
    Ret = Api_EnableMenuItem(Ret, SC_CLOSE, MF_BYCOMMAND Or MF_DISABLED Or MF_GRAYED)
'   Ret = Api_EnableMenuItem(Ret, SC_CLOSE, MF_BYCOMMAND Or MF_DISABLED)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    End
End Sub


'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop : End
```

閉じる「×」ボタンの無効化（IV）

閉じる「×」ボタンを無効化します。
GetSystemMenu システムメニューのハンドル取得
DeleteMenu システムメニューの項目を削除
DrawMenuBar メニューバーを再描画

起動直後

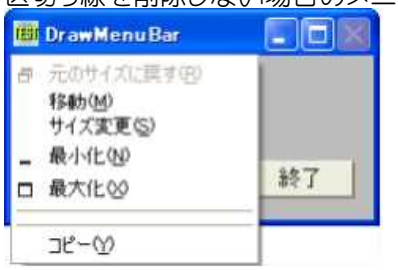


メニューの内容

「実行」ボタンで「×」を無効化

メニューの内容

区切り線を削除しない場合のメニューの内容

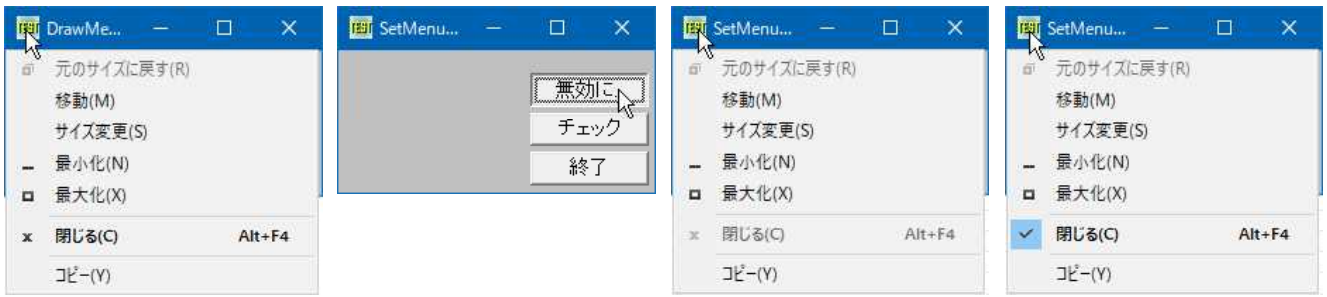



```
'=======================================
'= 閉じる「×」ボタンの無効化（IV）
'=    (DrawMenuBar.bas)
'=======================================
#include "Windows.bi"

' システムメニューのハンドル取得
Declare Function Api_GetSystemMenu& Lib "user32" Alias "GetSystemMenu" (ByVal hWnd&,
ByVal bRevert&)

' システムメニューの項目を削除
Declare Function Api_DeleteMenu& Lib "user32" Alias "DeleteMenu" (ByVal hMenu&, ByVal
nPosition&, ByVal wFlags&)

' メニューバーを再描画
Declare Function Api_DrawMenuBar& Lib "user32" Alias "DrawMenuBar" (ByVal hWnd&)

#define SC_CLOSE &HF060                    '閉じる
#define MF_BYCOMMAND &H0                   'nPositionはメニュー項目のID

Var Shared Button1 As Object
Var Shared Button2 As Object

Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14

'=======================================
'=
'=======================================
```

```
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hMenu As Long
    Var Ret As Long

    hMenu = Api_GetSystemMenu(GethWnd, 0)
    If hMenu Then

        '「×」を削除
        Ret = Api_DeleteMenu(hMenu, SC_CLOSE, MF_BYCOMMAND)

        '区切り線を削除
        Ret = Api_DeleteMenu(hMenu, 0, MF_BYCOMMAND)

        '変更を反映するために再描画
        Ret = Api_DrawMenuBar(GethWnd)
    End If
End Sub

'=======================================
'=
'=======================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    End
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

閉じる「×」ボタンの無効化とメニューチェック

GetSystemMenu システムメニューのハンドルを取得
GetMenuItemInfo メニュー項目に関する情報を取得
SetMenuItemInfo メニュー項目に関する情報を変更
SendMessage ウィンドウにメッセージを送信




```
'=======================================
'= 閉じる「×」ボタンの無効化とメニューチェック
'=    (SetmenuItemInfo.bas)
'=======================================
#include "Windows.bi"

#define SC_CLOSE &HF060                    '閉じる
#define MIIM_STATE &H1                     'fStateメンバをセット
#define MIIM_ID &H2                        'wIDメンバをセット
#define MFS_GRAYED &H3
#define MFS_CHECKED &H8
#define WM_NCACTIVATE &H86
```

```
Type MENUITEMINFO
    cbSize              As Long                 '構造体のバイト数
    fMask               As Long                 '取得する情報を指定する定数の組み合わせ
    fType               As Long                 'メニュー項目のタイプを指定する定数の組み合わせ
    fState              As Long                 'メニューの状態を指定する定数の組み合わせ
    wID                 As Long                 'ユーザー定義のメニュー項目のID
    hSubMenu            As Long                 '指定のメニュー項目と関連するサブメニューのハンドル
    hbmpChecked         As Long                 'チェックマーク用のビットマップのハンドル
    hbmpUnchecked       As Long                 '未チェック時のときのビットマップハンドル
    dwItemData          As Long                 'メニュー項目と関連するユーザー定義の値
    dwTypeData          As Long                 'メニュー項目のタイプ(fMaskにMIIM_Typeを指定したときのみ有効)
    cch                 As Long                 'メニュー項目のテキストのバイト数
End Type

' システムメニューのハンドル取得
Declare Function Api_GetSystemMenu& Lib "user32" Alias "GetSystemMenu" (ByVal hWnd&,
ByVal bRevert&)

' メニュー項目に関する情報を取得
Declare Function Api_GetMenuItemInfo& Lib "user32" Alias "GetMenuItemInfoA" (ByVal
hMenu&, ByVal uItem&, ByVal fByPosition&, lpMInfo As MENUITEMINFO)

' メニュー項目に関する情報を変更
Declare Function Api_SetMenuItemInfo& Lib "user32" Alias "SetMenuItemInfoA" (ByVal
hMenu&, ByVal uItem&, ByVal fByPosition&, lpMInfo As MENUITEMINFO)

' ウィンドウにメッセージを送信。
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

#define xSC_CLOSE -10
#define SwapID 1
#define ResetID 2

Var Shared Button(2) As Object
For i = 0 To 2
    Button(i).Attach GetDlgItem("Button" & Trim$(Str$(i + 1)))
    Button(i).SetFontSize 14
Next

Var Shared hMenu As Long
Var Shared mii As MENUITEMINFO

'=====================================
'=
'=====================================
Declare Function SetId(Action As Long) As Long
Function SetId(Action As Long) As Long
    Var MenuID As Long
    Var Ret As Long

    MenuID = mii.wID

    If mii.fState = (mii.fState Or MFS_GRAYED) Then
        If Action = SwapID Then
            mii.wID = SC_CLOSE
        Else
            mii.wID = xSC_CLOSE
        End If
    Else
        If Action = SwapID Then
            mii.wID = xSC_CLOSE
        Else
            mii.wID = SC_CLOSE
        End If
    End If

    mii.fMask = MIIM_ID
    Ret = Api_SetMenuItemInfo(hMenu, MenuID, False, mii)
```

```
    If Ret = 0 Then
        mii.wID = MenuID
    End If

    SetId = Ret
End Function

'=====================================
'=
'=====================================
Declare Sub SetButtons()
Sub SetButtons()
    If mii.fState = (mii.fState Or MFS_GRAYED) Then
        Button(0).SetWindowtext "有効に..."
    Else
        Button(0).SetWindowtext "無効に..."
    End If
    If mii.fState = (mii.fState Or MFS_CHECKED) Then
        Button(1).SetWindowtext "チェックなし"
    Else
        Button(1).SetWindowtext "チェック"
    End If
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var Ret As Long

    hMenu = Api_GetSystemMenu(GethWnd, 0)
    mii.cbSize = Len(MII)
    mii.dwTypeData = StrAdr(String$(80, Chr$(0)))
    mii.cch = Len(mii.dwTypeData)
    mii.fMask = MIIM_STATE
    mii.wID = SC_CLOSE
    Ret = Api_GetMenuItemInfo(hMenu, mii.wID, False, mii)
    SetButtons
    Button(2).SetWindowtext "終了"
End Sub

'=====================================
'=「×」ボタンの有効・無効
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    Ret = SetId(SwapID)

    If Ret <> 0 Then
        If mii.fState = (mii.fState Or MFS_GRAYED) Then
            mii.fState = mii.fState - MFS_GRAYED
        Else
            mii.fState = (mii.fState Or MFS_GRAYED)
        End If

        mii.fMask = MIIM_STATE
        Ret = Api_SetMenuItemInfo(hMenu, mii.wID, False, mii)

        If Ret = 0 Then
            Ret = SetId(ResetID)
        End If

        Ret = Api_SendMessage(GethWnd, WM_NCACTIVATE, True, 0)
        SetButtons
    End If
End Sub
```

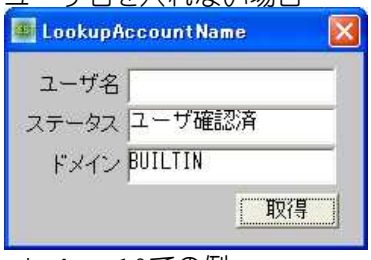
```
'=====================================
'= 閉じる「×」ボタンの無効化とチェック
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Ret As Long

    If mii.fState = (mii.fState Or MFS_CHECKED) Then
        mii.fState = mii.fState - MFS_CHECKED
    Else
        mii.fState = (mii.fState Or MFS_CHECKED)
    End If

    mii.fMask = MIIM_STATE

    Ret = Api_SetMenuItemInfo(hMenu, mii.wID, False, mii)
    SetButtons
End Sub

'=====================================
'=
'=====================================
Declare Sub Button3_on edecl ()
Sub Button3_on()
    End
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

ドメイン名の取得

ドメイン名を取得します。Windows9xは対象外、WindowsNT以降
LookupAccountName システム名とアカウントを入力として受け取る

WindowsXP(Valuestar)での取得例

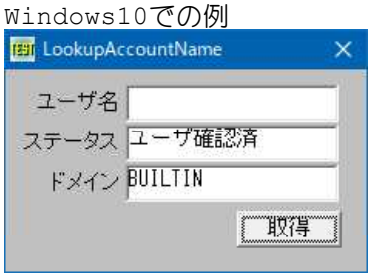
ユーザ名を入れない場合

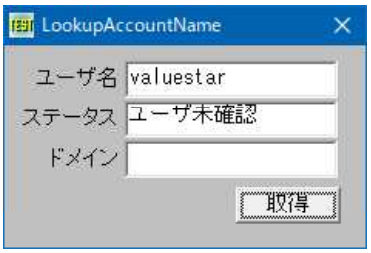


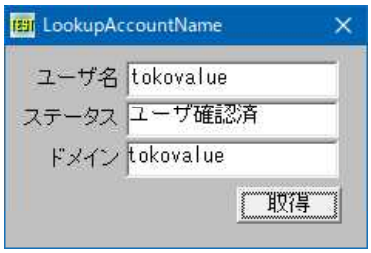
関係のない名前を入れた場合

当機のユーザ名を入れた場合

Windows10での例








```
'=====================================
'=ドメイン名の取得
'= WindowsNT/2000以降
'=    (LookupAccountName.bas)
'=====================================
```

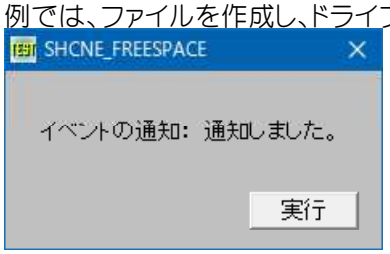
```
#include "Windows.bi"

' システム名とアカウントを入力として受け取る。そのアカウントのセキュリティ識別子(SID)と、アカウントが見つかったドメインの名前を取得
Declare Function Api_LookupAccountName& Lib "advapi32" Alias "LookupAccountNameA" (ByVal lpSystemName$, ByVal lpAccountName$, Sid As Byte, cbSid&, ByVal DomainName$, cbDomainName&, peUse&)

Var Shared Edit1 As Object
Var Shared Text(4) As Object
Var Shared Button1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
For i = 0 To 4
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1)))
    Text(i).SetFontSize 14
Next
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Function ValidateUser(sAccountName As String, sDomainName As String, sSystemName As String) As Integer
Function ValidateUser(sAccountName As String, sDomainName As String, sSystemName As String) As Integer
    Var success As Long
    Var cbSid As Long
    Var cbDomainName As Long
    Var peUse As Long

    sDomainName = Chr$(0)
    cbDomainName = 0

    If Len(sSystemName) = 0 Then
        sSystemName = Chr$(0)
    End If

    success = Api_LookupAccountName(sSystemName, sAccountName, 0, cbSid, sDomainName, cbDomainName, peUse)
    If cbSid = 0 Then Exit Function

    Var bSID(cbSid - 1) As Byte

    If (success = 0) And (cbSid > 0) Then
        sDomainName = Space$(cbDomainName)
        success = Api_LookupAccountName(sSystemName, sAccountName, bSID(0), cbSid, sDomainName, cbDomainName, peUse)
        If success > 0 Then
            If cbDomainName > 0 Then
                sDomainName = Left$(sDomainName, cbDomainName)
            End If
        End If
    End If
    ValidateUser = success
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var sAccount As String
    Var sSystem As String
    Var sDomain As String
    Var sValid As String
    Var vu As Integer

    sAccount = Edit1.GetWindowText
```

```
    sSystem = ""
    sDomain = ""

    vu = ValidateUser(sAccount, sDomain, sSystem)
    Select Case vu
        Case 1
            sValid = "ユーザ確認済"
        Case 0
            sValid = "ユーザ未確認"
    End Select

    Text(3).SetWindowText sValid
    Text(4).SetWindowText sDomain
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

ドライブの空き容量変化をシステムに通知

SHChangeNotify イベントをシステムに通知
SHCNE_FREESPACE(&H40000) ドライブに空きスペースが変化した
SHCNF_PATH(&H1) パス名SHCNF_PATHA
SHCNF_PATH(&H5) パス名SHCNF_PATHW(UNICODE)

例では、ファイルを作成し、ドライブの容量変化を通知した状態を示しています。




```
'=====================================
'=ドライブの空き容量変化をシステムに通知
'=    (SHChangeNotify.bas)
'=====================================
#include "Windows.bi"
#include "File.bi"

#define SHCNE_FREESPACE &H40000             'ドライブに空きスペースが変化した
#define SHCNF_PATH &H1                      'パス名SHCNF_PATHA
'#define SHCNF_PATH &H5                     'パス名(UNICODE)SHCNF_PATHW
' イベントをシステムに通知
Declare Sub Api_SHChangeNotify Lib "shell32" Alias "SHChangeNotify" (ByVal wEventId&,
ByVal uFlags&, dwItem1 As Any, dwItem2 As Any)

Var Shared Text1 As Object
Var Shared Text2 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Text2.Attach GetDlgItem("Text2") : Text2.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
```

```
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var FF As Long
    Var Drv As String

    '使用可能なファイル番号を取得
    FF = FreeFile

    'ファイルを作成
    Open GetWindowText & ".txt" For Output As FF

    'ファイルを閉じる
    Close FF

    'ドライブ文字を切り出し
    Drv = Left$(CrDir(""), 1)

    'イベントをシステムに通知
    Api_SHChangeNotify SHCNE_FREESPACE, SHCNF_PATH, Drv, ByVal CLng(0)

    '結果を表示
    Text2.SetWindowText "通知しました。"

    'ファイル作成･削除確認のため
    A% = MessageBox("ファイル作成･削除確認", "通知しました。", 0, 2)

    'ファイルを削除
    Kill GetWindowText & ".txt"
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

ドライブのファイル名最大長を取得

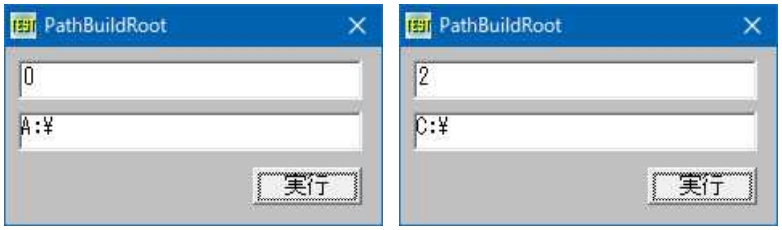
GetLogicalDrives 利用可能ディスクドライブ取得
GetVolumeInformation ルートディレクトリが呼び出しで指定されたファイルシステムとボリュームについての情報を返す


```
'=====================================
'=ドライブのファイル名最大長を取得
'=    (MaxComponentLength.bas)
'=====================================
#include "Windows.bi"

#define MAX_PATH 260

' 利用可能ディスクドライブ取得
Declare Function Api_GetLogicalDrives& Lib "kernel32" Alias "GetLogicalDrives" ()

' ルート ディレクトリが呼び出しで指定されたファイルシステムとボリュームについての情報を返す
Declare Function Api_GetVolumeInformation& Lib "Kernel32" Alias "GetVolumeInformationA"
```

```
(ByVal RootPathName$, ByVal VolNameBuff$, ByVal VolNameSize&, VolSerialNum&,
MaxComponentLen&, FileSysFlag&, ByVal FileSysNameBuff$, ByVal FileSysNameSize&)

Var Shared Text1 As Object
Var Shared Combo1 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var Shared Drive As String

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Drives = Api_GetLogicalDrives()         '利用可能なディスクドライブ取得
    If Drives = 0 Then Exit Sub             '関数の失敗

    For i = 0 To 25                         'A～Zドライブを検索する
        If (Drives And 1) = 1 Then
            Drive = Chr$(65 + i)            'ドライブ名(A～Z)に変換
            Drive = Drive & ":¥"
            Combo1.AddString Drive
        End If
        Drives = Drives ¥ 2                 'ドライブ検索
    Next i
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var PathName As String
    Var Buffer As String * MAX_PATH
    Var SerialNumber As Long
    Var MaxComponentLength As Long
    Var Flags As Long
    Var FileSystemName As String * 32
    Var Ret As Long

    '対象のルートパス名を指定
    PathName = Left$(Combo1.GetText(Combo1.GetCursel), 3)

    'ファイルシステムの情報を取得
    Ret = Api_GetVolumeInformation(PathName, Buffer, Len(Buffer), SerialNumber,
MaxComponentLength, Flags, FileSystemName, Len(FileSystemName))

    'ファイル名最大長を取得できたとき
    If MaxComponentLength Then
        Text1.SetWindowText Str$(MaxComponentLength)
    Else
        Text1.SetWindowText "ドライブ選択"
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

ドライブ番号からルートパスを生成

PathBuildRootドライブ番号からルートパスを生成

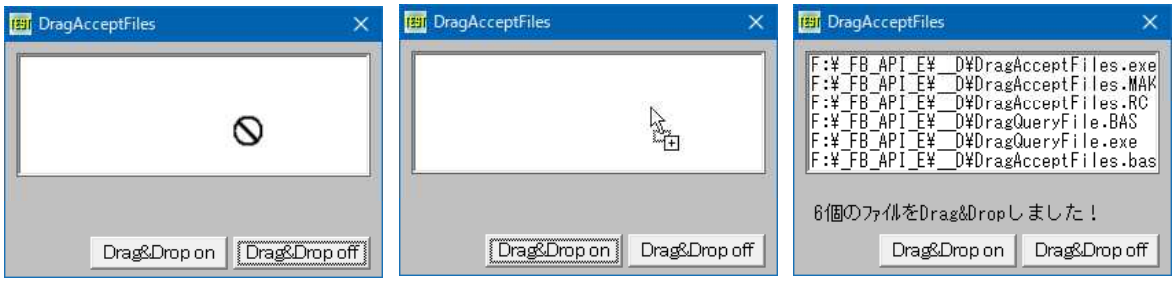



```
'=====================================
'=ドライブ番号からルートパスを生成
'=    (PathBuildRoot.bas)
'=====================================
#include "Windows.bi"

'ドライブ番号からルートパスを生成
Declare Function Api_PathBuildRoot& Lib "shlwapi" Alias "PathBuildRootA" (ByVal szRoot$,
ByVal iDrive$)

Var Shared Edit1 As Object
Var Shared Text1 As Object
Var Shared Button1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'= Chr$(0)を取り除く
'=====================================
Declare Function TrimNull (item As String) As String
Function TrimNull(item As String) As String
    Var ePos As Integer

    ePos = Instr(item, Chr$(0))
    If ePos Then
        TrimNull = Left$(item, ePos - 1)
    Else
        TrimNull = item
    End If
End Function

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Edit1.SetWindowText "0"
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var sSave As String
    Var Ret As Long

    sSave = Edit1.GetWindowText & String$(100, 0)

    Ret = Api_PathBuildRoot(sSave, ByVal Val(sSave))

    Text1.SetWindowText TrimNull(sSave)
End Sub
```

```
'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

ドラッグアンドドロップのON/OFF

DragAcceptFiles ウィンドウがファイルのドラッグアンドドロップを受け入れるかどうかを設定



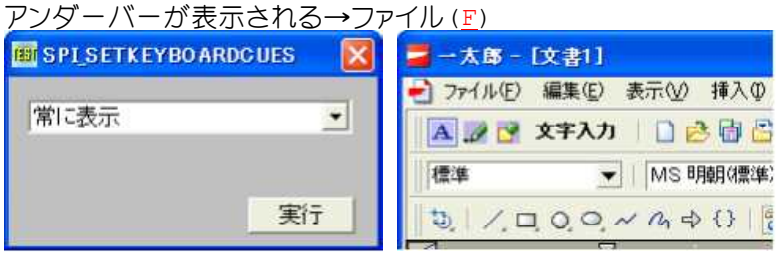

```
'=====================================
'=ドラッグアンドドロップのON/OFF
'=    (DragAcceptFiles.bas)
'=====================================
#include "Windows.bi"

' ウィンドウがファイルのドラッグアンドドロップを受け入れるかどうかを設定
Declare Sub Api_DragAcceptFiles Lib "shell32" Alias "DragAcceptFiles" (ByVal hWnd&,
ByVal fAccept&)

Var Shared Text1 As Object
Var Shared List1 As Object
Var Shared Button1 As Object
Var Shared Button2 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Api_DragAcceptFiles List1.GethWnd, True
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Api_DragAcceptFiles List1.GethWnd, False

    Text1.SetWindowText ""
    List1.ResetContent
End Sub

'=====================================
'= シェルドロップされたファイル名を取得
'=====================================
Declare Sub List1_DropFiles edecl (ByVal DF As Long)
```

```
Sub List1_DropFiles(ByVal DF As Long)
    Var FileName As String
    Var CN As Long
    Var i As Long

    List1.ResetContent

    CN = GetDropFileCount(DF)
    For i = 0 To CN - 1
        FileName = GetDropFileName(DF, i)
        List1.AddString FileName
    Next

    Text1.SetWindowText Str$(CN) & "個のﾌｧｲﾙをDrag&&Dropしました！"
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

ナビゲーションインジケータの有効無効を指定

SystemParametersInfo システム全体に関するパラメータを取得･設定
SPI_SETKEYBOARDCUES(&H100B) メニューアクセスキー文字の下線の設定
SPIF_SENDWININICHANGE(&H2) 全てのアプリケーションに通知して更新
SPIF_UPDATEINIFILE(&H1) ユーザープロファイルの更新を指定

アンダーバーが表示される→ファイル(F)



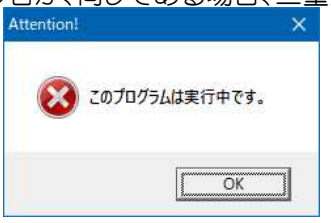
Altを押したときアンダーバーが表示される→ファイル(F)


```
'=======================================
'= ナビゲーションインジケータの有効無効を指定
'=    (SPI_SETKEYBOARDCUES.bas)
'=======================================
#include "Windows.bi"

' システム全体に関するパラメータを取得･設定
Declare Function Api_SystemParametersInfo& Lib "user32" Alias "SystemParametersInfoA"
(ByVal uiAction&, ByVal uiParam&, pvParam As Any, ByVal fWinIni&)

#define SPI_SETKEYBOARDCUES &H100B          '
#define SPIF_SENDWININICHANGE &H2           '全てのアプリケーションに通知して更新
#define SPIF_UPDATEINIFILE &H1              'ユーザープロファイルの更新を指定

Var Shared Combo1 As Object
Var Shared Button1 As Object
```

```
Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Combo1.AddString "Altを押さない時は非表示"
    Combo1.AddString "常に表示"
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Underline As Long
    Var Ret As Long

    'ナビゲーションインジケータの有効無効を指定
    Underline = Combo1.GetCursel

    'ナビゲーションインジケータを設定
    lngWin32apiResultCode = Api_SystemParametersInfo(SPI_SETKEYBOARDCUES, 0, ByVal
Underline, SPIF_UPDATEINIFILE Or SPIF_SENDCHANGE)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

二重起動の禁止(Ⅰ)

二つのアプリケーションのファイル名が、同じである場合、二重起動を禁止します。



Windows10での例

```
'=====================================
'= 二重起動の禁止(Ⅰ)
'=    (PrevInstance.bas)
'=====================================
If PrevInstance Then
    A% = MessageBox(GetWindowText, "このプログラムは実行中です。", 0, 0) : End
End If
```

二重起動の禁止(Ⅱ)

二つのアプリケーションのファイル名が、異なっていても、二重起動を禁止します。
OpenMutex 既存の名前付きミューテックスオブジェクトを開く
CreateMutex 名前付きまたは名前なしのミューテックスオブジェクトを作成または開く
ReleaseMutex 指定されたミューテックスオブジェクトの所有権を解放
CloseHandle オープンされているオブジェクトハンドルをクローズ

「Test Mutex」という名前のミューテックスが、あるかどうかを確認し、無ければ「Test Mutex」を作成して起動します。有る場合は、その旨メッセージを表示させ終了します。

初回起動時

もう1個起動した場合


```
'=====================================
'= 二重起動の防止
'=    (CreateMutex.bas)
'=====================================
#include "Windows.bi"

' 既存の名前付きミューテックスオブジェクトを開く
Declare Function Api_OpenMutex& Lib "kernel32" Alias "OpenMutexA" (ByVal
dwDesiredAccess&, ByVal bInheritHandle&, ByVal lpName$)

' 名前付きまたは名前なしのミューテックスオブジェクトを作成または開く
Declare Function Api_CreateMutex& Lib "kernel32" Alias "CreateMutexA" (ByRef
lpMutexAttributes As Any, ByVal bInitialOwner&, ByVal lpName$)

' 指定されたミューテックスオブジェクトの所有権を解放
Declare Function Api_ReleaseMutex& Lib "kernel32" Alias "ReleaseMutex" (ByVal hMutex&)

' オープンされているオブジェクトハンドルをクローズ
Declare Function Api_CloseHandle& Lib "Kernel32" Alias "CloseHandle" (ByVal hObject&)

#define ERROR_ALREADY_EXISTS 183          '既に存在している
#define MUTEX_ALL_ACCESS &H1F0001         '

Var Shared Mutex As Long

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var Ret As Long
    'ミューテックスオブジェクトを開く
    Mutex = Api_OpenMutex(MUTEX_ALL_ACCESS, 0, "Test Mutex")

    If Mutex = 0 Then

        'ミューテックスオブジェクトを作成または開く
        Mutex = Api_CreateMutex(ByVal 0, 0, "Test Mutex")

        A% = MessageBox(GetWindowText, "起動しました！", 0, 2)
    Else

        'オブジェクトハンドルをクローズ
        Ret = Api_CloseHandle(Mutex)

        A% = MessageBox(GetWindowText, "既に起動しています！", 0, 2)

        'ミューテックスオブジェクトの所有権を解放
        Ret = Api_ReleaseMutex(Mutex)
        End
    End If
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
```

```
    Var Ret As Long

    'ミューテックスオブジェクトの所有権を解放
    Ret = Api_ReleaseMutex(Mutex)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

入力する文字数の制限

入力する文字数を制限します。F-BASICのSETLIMITTEXTと同じです。
SendMessage ウィンドウにメッセージを送信
CB_LIMITTEXT(&H141) コンボボックスのエディットコントロール内のテキストの文字数を制限する
EM_LIMITTEXT(&HC5) エディットコントロール内のテキストの文字数を制限する

リミットに制限する文字数を入力し「制限」ボタンをクリック後、EditBoxまたはComboBoxに文字を入力しています。
「解除」ボタンでデフォルトに設定されます。

5文字に制限例

15文字に制限例

制限解除


```
'=====================================
'= 入力する文字数の制限
'=    (LIMITTEXT.bas)
'=====================================
#include "Windows.bi"

#define CB_LIMITTEXT &H141                  'コンボボックスのエディットコントロール内のテキストの文字数を制限
#define EM_LIMITTEXT &HC5                   'エディットコントロール内のテキストの文字数を制限する

' ウィンドウにメッセージを送信。この関数は、指定したウィンドウのウィンドウプロシージャが処理を終了するまで制御を返さない
Declare Function Api_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal wMsg&, ByVal wParam&, lParam As Any)

' ウインドウのタイトルを変更
Declare Function Api_SetWindowText& Lib "user32" Alias "SetWindowTextA" (ByVal hWnd&, ByVal lpString$)

Var Shared Text1 As Object
Var Shared Combo1 As Object
Var Shared Edit(1) As Object
Var Shared Button(1) As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
For i = 0 To 1
    Edit(i).Attach GetDlgItem("Edit" & Trim$(Str$(i + 1))) : Edit(i).SetFontSize 14
    Button(i).Attach GetDlgItem("Button" & Trim$(Str$(i + 1))) : Button(i).SetFontSize 14
Next i

'=====================================
'=
'=====================================
```

```
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var maxTxtLen As Long
    Var Ret As Long

    'リミット文字数取得
    maxTxtLen = CLng(val(Edit(1).GetWindowText))

    'テキストクリア
    Ret = Api_SetWindowText(Edit(0).GethWnd, "")
    Ret = Api_SetWindowText(Combo1.GethWnd, "")

    'リミット設定
    Ret = Api_SendMessage(Edit(0).GethWnd, EM_LIMITTEXT, maxTxtLen, ByVal 0)
    Ret = Api_SendMessage(Combo1.GethWnd, CB_LIMITTEXT, maxTxtLen, ByVal 0)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Ret As Long

    'テキストクリア
    Ret = Api_SetWindowText(Edit(0).GethWnd, "")
    Ret = Api_SetWindowText(Edit(1).GethWnd, "")
    Ret = Api_SetWindowText(Combo1.GethWnd, "")

    'リセット
    Ret = Api_SendMessage(Edit(0).GethWnd, EM_LIMITTEXT, 0, ByVal 0)
    Ret = Api_SendMessage(Combo1.GethWnd, CB_LIMITTEXT, 0, ByVal 0)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

ネットワークインターフェース一覧の取得

ネットワークインターフェース一覧を取得します。
CopyMemory メモリブロックの移動
GetIpAddrTable ネットワークインターフェース一覧の取得(Windows98以降)


```
'=====================================
'= ネットワークインターフェース一覧の取得
'=    (GetIpAddrTable.bas)
'=====================================
#include "Windows.bi"

#define MAX_IP 4                        'とりあえず5個以内のIPアドレスを想定
```

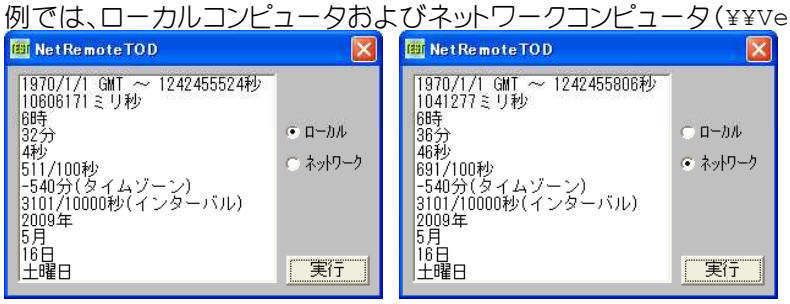
```
Type IPINFO
    dwAddr            As Long               'IP アドレス
    dwIndex           As Long               'インタフェースインデックス
    dwMask            As Long               'IPアドレスのうちネットワークアドレスとホストアドレスを識別
                                            するための数値
    dwBCastAddr       As Long               'ネットワーク内のすべての端末にデータを送信するために
                                            使われる特殊なアドレス
    dwReasmSize       As Long               'アセンブリサイズ
    unused1           As Integer            '未使用
    unused2           As Integer            '未使用
End Type

Type MIB_IPADDRTABLE
    dEntrys           As Long               'エントリ数
    mIPInfo(MAX_IP)   As IPINFO             'IPアドレスエントリーの配列
End Type

Type IP_Array
    mBuffer    As MIB_IPADDRTABLE
    BufferLen  As Long
End Type

' ある位置から別の位置にメモリブロックを移動する関数の宣言
Declare Sub CopyMemory Lib "Kernel32" Alias "RtlMoveMemory" (Destination As Any, Source
As Any, ByVal Length&)

' ネットワークインターフェース一覧の取得
Declare Function Api_GetIpAddrTable& Lib "Iphlpapi" Alias "GetIpAddrTable" (pIPAdrTable
As Byte, pdwSize&, ByVal Sort&)

Var Shared List1 As Object

List1.Attach GetDlgItem("List1") : List1.SetFontSize 12

'=====================================
'=
'=====================================
Declare Function ConvertAddressToString(longAddr As Long) As String
Function ConvertAddressToString(longAddr As Long) As String
    Var myByte(3) As Byte
    Var Str As String
    Var Cnt As Long

    CopyMemory myByte(0), longAddr, 4
    For Cnt = 0 To 3
        Str = Str & Right$("   " + Str$(myByte(Cnt)), 3) & "."
    Next Cnt

    Str = Left$(Str, Len(Str) - 1)
    ConvertAddressToString = Str
End Function

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var ip As Long
    Var mi As MIB_IPADDRTABLE
    Var Ret As Long

    Ret = Api_GetIpAddrTable(ByVal 0, Ret, True)

    If Ret <= 0 Then Exit Sub

    Var bBytes(Ret - 1) As Byte

    Ret = Api_GetIpAddrTable(bBytes(0), Ret, False)

    CopyMemory mi.dEntrys, bBytes(0), 4
```

```
    List1.Resetcontent
    List1.AddString Str$(mi.dEntrys) & " IP Address が見つかりました！"
    List1.AddString "----------------------------------------"
    For ip = 0 To mi.dEntrys - 1
        CopyMemory mi.mIPInfo(ip), bBytes(4 + (ip * Len(mi.mIPInfo(0)))), Len(mi.mIPInfo
(ip))
        List1.AddString "IP Address          : " &
ConvertAddressToString(mi.mIPInfo(ip).dwAddr)
        List1.AddString "IP Subnetmask       : " &
ConvertAddressToString(mi.mIPInfo(ip).dwMask)
        List1.AddString "BroadCast IP Address : " &
ConvertAddressToString(mi.mIPInfo(ip).dwBCastAddr)
        List1.AddString "----------------------------------------"
    Next
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

ネットワークコンピュータの日付・時間を取得

NetRemoteTOD サーバーの時間を取得
NetApiBufferFree バッファの解放
CopyMemory ある位置から別の位置にメモリブロックを移動
MultiByteToWideChar ANSI文字列をUnicode文字列に変換

例では、ローカルコンピュータおよびネットワークコンピュータ(¥¥VersaPro)の日付・時間を取得しています。




```
'=====================================
'= ネットワークコンピュータの日付・時間を取得
'=     (NetRemoteTOD.bas)
'=====================================
#include "Windows.bi"

' サーバーの時間を取得
Declare Function Api_NetRemoteTOD& Lib "Netapi32" Alias "NetRemoteTOD" (UncServerName As
Any, BufferPtr&)

' バッファの解放
Declare Function Api_NetApiBufferFree& Lib "netapi32" Alias "NetApiBufferFree"
(lpbyteptr&)

' ある位置から別の位置にメモリブロックを移動
Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (hpvDest As Any, ByVal
hpvSource&, ByVal cbCopy&)

' ANSI文字列をUnicode文字列に変換
Declare Function Api_MultiByteToWideChar& Lib "Kernel32" Alias "MultiByteToWideChar"
(ByVal CodePage&, ByVal dwFlags&, ByVal lpMultiByteStr$, ByVal cchMultiByte&, ByVal
lpWideCharStr$, ByVal cchWideChar&)
```

```
#define CP_ACP 0                                'ANSIコードページ
#define CP_MACCP 2                              'Macintoshコードページ
#define CP_OEMCP 1                              'OEMコードページ
#define CP_SYMBOL 42                            'シンボルコードページ(Windows2000･XP)
#define CP_THREAD_ACP 3                         '呼び出しスレッドのANSIコードページ(Windows2000･XP)
#define CP_UTF7 65000                           'UTF-7を使用して変換(Windows98･Me･NT4.0以降)
#define CP_UTF8 65001                           'UTF-8を使用して変換(Windows98･Me･NT4.0以降)これ
                                                を指定した場合、dwFlagsパラメータは0

Type TIME_OF_DAY_INFO
    tod_elapsedt      As Long
    tod_msecs         As Long                   '起動からの経過
    tod_hours         As Long
    tod_mins          As Long
    tod_secs          As Long
    tod_hunds         As Long                   '1/100
    tod_timezone      As Long
    tod_tinterval     As Long
    tod_day           As Long
    tod_month         As Long
    tod_year          As Long
    tod_weekday       As Long
End Type

Var Shared List1 As Object
Var Shared Radio1 As Object
Var Shared Radio2 As Object
Var Shared Button1 As Object

List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
Radio1.Attach GetDlgItem("Radio1") : Radio1.SetFontSize 14
Radio2.Attach GetDlgItem("Radio2") : Radio2.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Function Index bdecl () As Integer
Function Index()
    Index = Val(Mid$(GetDlgRadioSelect("Radio1"), 6)) - 1
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var todi As TIME_OF_DAY_INFO
    Var Buff As String                          '元の文字列
    Var wBuff As String                         '変換先文字列
    Var wBuffLen As Long                        '変換先文字数
    Var BufPtr As Long
    Var Ret As Long

    'サーバー名を設定
    Select Case Index

        'ローカルコンピュータの場合
        Case 0
            Buff = Chr$(0)

        'ネットワークコンピュータの場合(エラー処理なし)
        Case 1
            Buff = "¥¥VersaPro" & Chr$(0)
    End Select

    wBuff = String$(256, Chr$(0))

    'Unicodeに変換
    wBuffLen = Api_MultiByteToWideChar(CP_ACP, 0, Buff, -1, wBuff, 0)
    Ret = Api_MultiByteToWideChar(CP_ACP, 0, Buff, -1, wBuff, wBuffLen)
```

```
    wBuff = Left$(wBuff, (wBuffLen - 1) * 2)

    'リモートコンピュータのシステム時間を取得
    Ret = Api_NetRemoteTOD(wBuff, BufPtr)

    'バッファを構造体にコピー
    CopyMemory todi, ByVal BufPtr, Len(todi)

    'メモリを解放
    If BufPtr <> 0 Then
        Ret = Api_NetApiBufferFree(BufPtr)
    End If

    List1.Resetcontent

    List1.AddString "1970/1/1 GMT ～ " & Trim$(Str$(todi.tod_elapsedt)) & "秒"
    List1.AddString Trim$(Str$(todi.tod_msecs)) & "ミリ秒"
    List1.AddString Trim$(Str$(todi.tod_hours)) & "時"
    List1.AddString Trim$(Str$(todi.tod_mins)) & "分"
    List1.AddString Trim$(Str$(todi.tod_secs)) & "秒"
    List1.AddString Trim$(Str$(todi.tod_hunds)) & "1/100秒"
    List1.AddString Trim$(Str$(todi.tod_timezone)) & "分(タイムゾーン)"
    List1.AddString Trim$(Str$(todi.tod_tinterval)) & "1/10000秒(インターバル)"
    List1.AddString Trim$(Str$(todi.tod_year)) & "年"
    List1.AddString Trim$(Str$(todi.tod_month)) & "月"
    List1.AddString Trim$(Str$(todi.tod_day)) & "日"
    List1.AddString KMid$("日月火水木金土", todi.tod_weekday + 1, 1) & "曜日"
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop : End
```

ネットワーク上にフォルダを作成

SHBrowseForFolder「フォルダの参照」ダイアログを開く
SHGetPathFromIDList アイテムIDリストをファイルシステムのパス名に変換
lstrcat ある文字列の末尾に別の文字列を結合

エディットボックスにフォルダ名を入れ、「フォルダの参照」を開き、作成する親フォルダを指定します。


```
'=====================================
'= ネットワーク上にフォルダを作成
'=    (SHBrowseForFolder3.bas)
'=====================================
#include "Windows.bi"
#include "File.bi"

Type BROWSEINFO
    hWndOwner         As Long
    pIDLRoot          As Long
    pszDisplayName    As Long
    lpszTitle         As Long
    ulFlags           As Long
    lpfnCallback      As Long
    lParam            As Long
    iImage            As Long
End Type
```

```
'「フォルダの参照」ダイアログを開く
Declare Function Api_SHBrowseForFolder& Lib "shell32" Alias "SHBrowseForFolder" (lpbi As
BROWSEINFO)

' アイテムIDリストをファイルシステムのパス名に変換
Declare Function Api_SHGetPathFromIDList& Lib "shell32" Alias "SHGetPathFromIDList"
(ByVal pidList&, ByVal lpBuffer$)

' ある文字列の末尾に別の文字列を結合
Declare Function Api_lstrcat& Lib "Kernel32" Alias "lstrcatA" (ByVal lpString1$, ByVal
lpString2$)

#define BIF_RETURNONLYFSDIRS &H1            'ファイルシステムディレクトリのみを返す
#define CSIDL_NETWORK &H12                  'ネットワークコンピュータ(仮想フォルダ)
#define MAX_PATH 260                        '

Var Shared Edit1 As Object
Var Shared Text1 As Object
Var Shared Button1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var pidList As Long
    Var Buffer As String
    Var bi As BROWSEINFO
    Var NetPath As String
    Var NewFolder As String
    Var Ret As Long
    bi.hWndOwner = GethWnd
    bi.pIDLRoot = CSIDL_NETWORK
    bi.ulFlags = BIF_RETURNONLYFSDIRS
    bi.lpszTitle = Api_lstrcat("ネットワークの参照", "")

    pidList = Api_SHBrowseForFolder(bi)
    If pidList Then
        Buffer = Space$(MAX_PATH)
        Ret = Api_SHGetPathFromIDList(pidList, Buffer)
        If Ret Then
            NetPath = Left$(Buffer, InStr(Buffer, Chr$(0)) - 1)
            Text1.SetWindowText NetPath
        Else
            Text1.SetWindowText "パス取得に失敗しました。"
            Exit Sub
        End If
    Else
        Text1.SetWindowText "失敗しました"
        Exit Sub
    End If

    '作成するフォルダ名
    NewFolder = NetPath & "¥" & Edit1.GetWindowText

    'フォルダを作成
    MkDir NewFolder
    Text1.SetWindowText NewFolder & "を作成しました。"
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
```

```
Stop
End
```

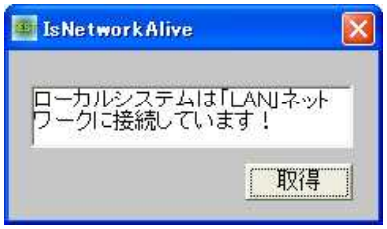
ネットワーク接続状態と種類（Ⅰ）

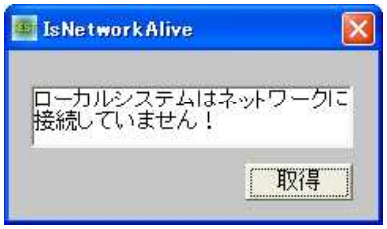
ローカルシステムがネットワークに接続されているか、接続されている場合はその種類を取得します。
IsNetworkAlive ネットワーク接続状態を取得


```
'=====================================
'= ネットワーク接続状態と種類
'=    (IsNetworkAlive2.bas)
'=====================================

' ローカルシステムがネットワークおよびネットワークに接続しているかどうかを決定
Declare Function Api_IsNetworkAlive& Lib "Sensapi" Alias "IsNetworkAlive" (ByRef
lpdwFlags&)

#define NETWORK_ALIVE_AOL &H4              'AOL接続(Windows9x)
#define NETWORK_ALIVE_LAN &H1              'LAN(Local Area Network)接続
#define NETWORK_ALIVE_WAN &H2              'WAN(Wide Area Network)接続

Var Net As String
Var Ret As Long

If Api_IsNetworkAlive(Ret) = 0 Then
    Print "ローカルシステムはネットワークに接続されていません！"
Else
    If Ret = NETWORK_ALIVE_AOL Then
        Net = "AOL"
    Else
        If Ret = NETWORK_ALIVE_LAN Then
           Net = "LAN"
        Else
           Net = "WAN"
        End If
    End If

    Print "ローカルシステムは " & Net & " に接続されています！"
End If

Stop
End
```

ネットワークの接続と種類（Ⅱ）

ネットワークの接続状態とその種類を取得します。
IsNetworkAlive ローカルシステムがネットワークおよびネットワークに接続しているかどうかを決定





```
'=====================================
'= ネットワークの接続と種類
'=    (IsNetworkAlive.bas)
'=====================================
#include "Windows.bi"

' ローカルシステムがネットワークおよびネットワークに接続しているかどうかを決定
Declare Function Api_IsNetworkAlive& Lib "Sensapi" Alias "IsNetworkAlive" (ByRef
lpdwFlags&)

#define NETWORK_ALIVE_AOL &H4              'AOL接続(Windows9x)
#define NETWORK_ALIVE_LAN &H1              'LAN(Local Area Network)接続
#define NETWORK_ALIVE_WAN &H2              'WAN(Wide Area Network)接続

Var Shared Text1 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Flg As Long

    If Api_IsNetworkAlive(Flg) = 0 Then
        Text1.SetWindowText "ローカルシステムはネットワークに接続していません！"
    Else
        If Flg = NETWORK_ALIVE_AOL Then

            Text1.SetWindowText "ローカルシステムは「AOL」ネットワークに接続しています！"
        Else If Flg = NETWORK_ALIVE_LAN Then
            Text1.SetWindowText "ローカルシステムは「LAN」ネットワークに接続しています！"
        Else If Flg = NETWORK_ALIVE_WAN Then

            Text1.SetWindowText "ローカルシステムは「WAN」ネットワークに接続しています！"
        End If
    End If
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

ネットワーク接続の名前を取得

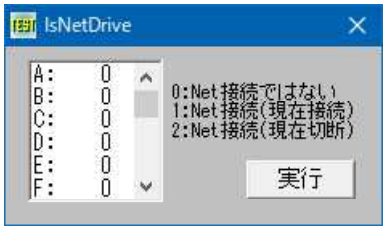
GetDriveType ドライブのタイプを取得
WNetGetConnection ローカル装置に対応するネットワーク資源の名前を取得

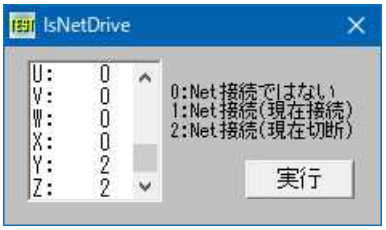
```
'=====================================
'= ネットワーク接続の名前を取得
'=    (WNetGetConnection.bas)
'=====================================
#include "Windows.bi"

'ドライブのタイプを取得
Declare Function Api_GetDriveType& Lib "Kernel32" Alias "GetDriveTypeA" (ByVal nDrive$)

'ローカル装置に対応するネットワーク資源の名前を取得
Declare Function Api_WNetGetConnection& Lib "mpr" Alias "WNetGetConnectionA" (ByVal
lpszLocalName$, ByVal lpszRemoteName$, cbRemoteName&)

#define DRIVE_CDROM 5                       'CD-ROMドライブ
#define DRIVE_FIXED 3                       '固定タイプ－主にハードディスク
#define DRIVE_NO_ROOT_DIR 1                 'ルートディレクトリ無し
#define DRIVE_RAMDISK 6                     'RAMドライブ
#define DRIVE_REMOTE 4                      'ネットワーク
#define DRIVE_REMOVABLE 2                   '取り外し可能タイプ－リムーバブルディスク
#define DRIVE_UNKNOWN 0                     'ドライブが不明

#define WN_NO_ERROR  0                      '関数成功
#define WN_BAD_DEVICE 1200                  'lpLocalNameパラメータが指す文字列が無効
#define WN_CONNECTION_UNAVAILABLE 1201      '装置は現在接続されていないが、恒久的な接続として記憶
されている
#define WN_EXTENDED_ERROR 1208              'ネットワーク固有のエラーが発生
#define WN_MORE_DATA 234                    'バッファのサイズが不十分
#define WN_NOT_SUPPORTED 50                 'サポート外
#define WN_NO_NET_OR_BAD_PATH 1203          '指定したローカル名を使った接続を認識するプロバイダが
ない
#define WN_NO_NETWORK 1222                  'ネットワークに接続されていない
#define WN_NOT_CONNECTED 2250               'lpLocalNameパラメータで指定した装置がリダイレクトさ
れていない

Var Shared Text1 As Object
Var Shared Combo1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Function GetUNCName(Drive As String) As String
Function GetUNCName(Drive As String) As String
    Var Buffer As String
    Var Leng As Long
    Var Msg As String
    Var Ret As Long

    Buffer = Space$(255)
    Leng = Len(Buffer)
    Ret = Api_WNetGetConnection(Drive, Buffer, Leng)

    Select Case Ret
        Case WN_BAD_DEVICE
            Msg = "パラメータが指す文字列が無効"
        Case WN_CONNECTION_UNAVAILABLE
            Msg = "装置は現在接続されていないが、恒久的な接続として記憶されている"
        Case WN_EXTENDED_ERROR
            Msg = "ネットワーク固有のエラーが発生"
        Case WN_MORE_DATA
            Msg = "バッファのサイズが不十分"
        Case WN_NOT_SUPPORTED
            Msg = "サポート外"
        Case WN_NO_NET_OR_BAD_PATH
            Msg = "指定したローカル名を使った接続を認識するプロバイダがない"
        Case WN_NO_NETWORK
            Msg = "ネットワークに接続されていない"
        Case WN_NOT_CONNECTED
            Msg = "lpLocalNameパラメータで指定した装置がリダイレクトされていない"
```

```
        Case Else
            Msg = ""
    End Select

    If Msg <> "" Then
        A% = MessageBox("", Msg, 0, 2)
    Else
        GetUNCName = Left$(Buffer, Leng)
    End If
End Function

'=======================================
'=
'=======================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var i As Integer
    Var Drive As String * 2
    For i = 0 To 25
        Drive = Chr$(i + 65) & ":"
        Select Case Api_GetDriveType(Drive)
            Case DRIVE_REMOTE
                Combo1.AddString Drive
            Case Else
        End Select
    Next i
End Sub

'=======================================
'=
'=======================================
Declare Sub Combo1_Change edecl ()
Sub Combo1_Change()
  Text1.SetWindowText GetUNCName(Combo1.GetWindowText)
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

ネットワークドライブの判定

IsNetDrive ネットワークドライブの判定
(WindowsXP SP2、Windowsサーバー2003)
Windows2000以前では使えません。(Windows2000では、IsNetDriveでエラーになりました。GetDriveTypeまたはWNetGetConnectionを使います)






```
'=======================================
'= ネットワークドライブの判定
'= WindowsXP(SP2)、Windowsサーバー2003
'= OSによっては、GetDriveTypeまたはWNetGetConnectionを使う
'=    (IsNetDrive.bas)
'=======================================
```

```
#include "Windows.bi"

' ネットワークドライブの判定
Declare Function Api_IsNetDrive& Lib "Shell32" Alias "IsNetDrive" (ByVal iDrive&)

#define vbCrLf (Chr$(13) & Chr$(10))        'キャリッジリターンとラインフィード(¥r¥n)

Var Shared List1 As Object
Var Shared Text1 As Object
Var Shared Button1 As Object

List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 12
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var msg As String
    msg = msg & "0:Net接続ではない" & vbCrLf
    msg = msg & "1:Net接続(現在接続)" & vbCrLf
    msg = msg & "2:Net接続(現在切断)"

    Text1.SetWindowtext msg
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var i As Integer

    For i = 0 To 25
        List1.AddString Chr$(65 + i) & ":   " & Str$(Api_IsNetDrive(i))
    Next
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

ネットワークドライブのファイル名から付加情報を取得

ネットワークドライブ名から付加情報を取得します。
WNetGetUniversalNameドライブベースのパスをネットワーク資源とし、その資源のより汎用的な名前を含む情報構造体を取得

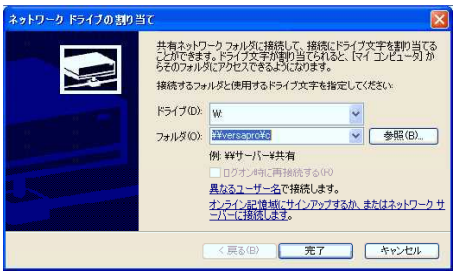
例では、XおよびYドライブの情報を取得しています。

```
'====================================
'= ネットワークドライブのファイル名から付加情報を取得
'=    (WNetGetUniversalName.bas)
'====================================
#include "Windows.bi"

'ドライブベースのパスをネットワーク資源とし、その資源のより汎用的な名前を含む情報構造体を取得
Declare Function Api_WNetGetUniversalName& Lib "mpr" Alias "WNetGetUniversalNameA"
(ByVal lpLocalPath$, ByVal dwInfoLevel&, lpBuffer As Any, lpBufferSize&)

' ある位置から別の位置にメモリブロックを移動する関数の宣言
Declare Sub CopyMemory Lib "Kernel32" Alias "RtlMoveMemory" (Destination As Any, Source
As Any, ByVal Length&)

Type REMOTE_NAME_INFO
    lpUniversalName   As Long
    lpConnectionName  As Long
    lpRemainingPath   As Long
End Type

#define UNIVERSAL_NAME_INFO_LEVEL &H1      'UNIVERSAL_NAME_INFO構造体をバッファに格納
#define REMOTE_NAME_INFO_LEVEL &H2         'REMOTE_NAME_INFO構造体をバッファに格納
#define NO_ERROR 0
Var Shared Text(6) As Object
Var Shared Edit1 As Object
Var Shared Button1 As Object

For i = 0 To 6
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1)))
    Text(i).SetFontSize 14
Next
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'====================================
'=
'====================================
Declare Sub MainForm_Start edecl ()
Sub mainForm_Start()
    Edit1.SetWindowtext "X:¥Windows¥Twein.dll"
End Sub

'====================================
'=
'====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var rni As REMOTE_NAME_INFO
    Var BufferSize As Long
    Var Buffer As String
    Var bytBuffer(1023) As Byte
    Var Ret As Long

    'リモート名情報を取得
    Ret = Api_WNetGetUniversalName(Edit1.GetWindowText, REMOTE_NAME_INFO_LEVEL,
bytBuffer(0), 1023)

    If Ret = NO_ERROR Then

        'バッファに取得した情報をリモート名情報構造体へコピー
        CopyMemory rni, bytBuffer(0), Len(rni)

        'UNC名を文字列バッファへコピー
        Buffer = Space$(511)
        CopyMemory Buffer, ByVal rni.lpUniversalName, Len(Buffer)

        'UNC名を表示
        Text(4).SetWindowText Left$(Buffer, InStr(Buffer, Chr$(0)) - 1)
```

```
        '接続名を文字列バッファへコピー
        Buffer = Space$(511)
        CopyMemory Buffer, ByVal rni.lpConnectionName, Len(Buffer)

        '接続名を文字列バッファへコピー
        Text(5).SetWindowText Left$(Buffer, InStr(Buffer, Chr$(0)) - 1)

        'パス名を文字列バッファへコピー
        Buffer = Space$(511)
        CopyMemory Buffer, ByVal rni.lpRemainingPath, Len(Buffer)

        'パス名を表示
        Text(6).SetWindowText Left$(Buffer, InStr(Buffer, Chr$(0)) - 1)
    End If
End Sub

'========================================
'=
'========================================
Declare Sub Edit1_SetFocus edecl ()
Sub Edit1_SetFocus()
    For i = 4 To 6
        Text(i).SetWindowtext ""
    Next
End Sub
'========================================
'=
'========================================
While 1
    WaitEvent
Wend
Stop
End
```

ネットワークドライブの割り当てを呼び出す

WNetConnectionDialog1 ネットワーク資源に接続するための汎用の参照ダイアログボックスを起動
WNetDisconnectDialog1 ネットワーク資源からの切断を試みる
GlobalAlloc メモリブロックを確保しハンドルを取得
GlobalFree メモリブロックのロックを解放
CopyMemory ある位置から別の位置にメモリブロックを移動



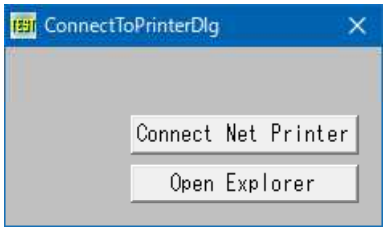

```
'========================================
'= ネットワークドライブの割り当てを呼び出す
'=    (WNetConnectionDialog1.bas)
'========================================
#include "Windows.bi"

#define CONNDLG_RO_PATH &H1
#define CONNDLG_CONN_POINT &H2
#define CONNDLG_USE_MRU &H4
#define CONNDLG_HIDE_BOX &H8
#define CONNDLG_PERSIST &H10
#define CONNDLG_NOT_PERSIST &H20

#define DISC_UPDATE_PROFILE &H1
#define DISC_NO_FORCE &H40
```

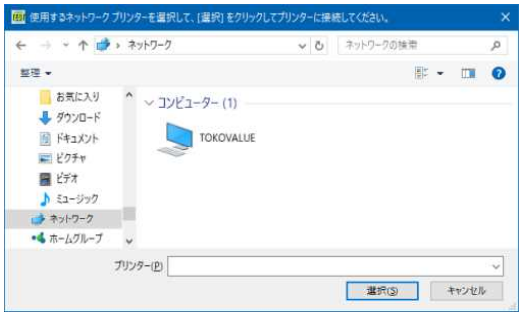
```
#define RESOURCETYPE_DISK &H1
#define NO_ERROR 0
#define WN_SUCCESS NO_ERROR

#define GMEM_FIXED &H0
#define GMEM_ZEROINIT &H40
#define GPTR (GMEM_FIXED Or GMEM_ZEROINIT)

Type CONNECTDLGSTRUCT
    cbStructure       As Long
    hwndOwner         As Long
    lpConnRes         As Long
    dwFlags           As Long
    dwDevNum          As Long
End Type

Type DISCDLGSTRUCT
    cbStructure       As Long
    hwndOwner         As Long
    lpLocalName       As Long
    lpRemoteName      As Long
    dwFlags           As Long
End Type

Type NETRESOURCE
    dwScope           As Long
    dwType            As Long
    dwDisplayType     As Long
    dwUsage           As Long
    lpLocalName       As Long
    lpRemoteName      As Long
    lpComment         As Long
    lpProvider        As Long
End Type

' ネットワーク資源に接続するための汎用の参照ダイアログボックスを起動
Declare Function Api_WNetConnectionDialog1& Lib "mpr" Alias "WNetConnectionDialog1A"
(lpConnectDlgStruct As Any)

' ネットワーク資源からの切断を試みる
Declare Function Api_WNetDisconnectDialog1& Lib "mpr" Alias "WNetDisconnectDialog1A"
(lpDiscDlgStruct As Any)

' メモリブロックを確保しハンドルを取得
Declare Function Api_GlobalAlloc& Lib "kernel32" Alias "GlobalAlloc" (ByVal wFlags&,
ByVal dwBytes&)

' メモリブロックのロックを解放
Declare Function Api_GlobalFree& Lib "kernel32" Alias "GlobalFree" (ByVal hMem&)

' ある位置から別の位置にメモリブロックを移動
Declare Sub CopyMemory Lib "Kernel32" Alias "RtlMoveMemory" (Destination As Any, Source
As Any, ByVal Length&)

Var Shared Text1 As Object
Var Shared Button1 As Object
Var Shared Button2 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var cs As CONNECTDLGSTRUCT
    Var nr As NETRESOURCE
    Var Ret As Long
```

```
    nr.lpRemoteName = StrAdr("¥¥versapro¥c" & Chr$(0))
    nr.dwType = RESOURCETYPE_DISK

    cs.cbStructure = Len(cs)
    cs.hwndOwner = GethWnd
    cs.lpConnRes = Api_GlobalAlloc(GPTR, Len(nr))

    CopyMemory ByVal cs.lpConnRes, nr, Len(nr)

    cs.dwFlags = CONNDLG_USE_MRU Or CONNDLG_HIDE_BOX Or CONNDLG_NOT_PERSIST

    Ret = Api_WNetConnectionDialog1(cs)

    If Ret = WN_SUCCESS Then
        A% = MessageBox("", "MapDrive Succeeded.", 0, 2)
        Text1.SetWindowText Chr$(cs.dwDevNum + 64) & ":"
    Else
        A% = MessageBox("", "Error!", 0, 2)
    End If
    Ret = Api_GlobalFree(cs.lpConnRes)
End Sub

'=======================================
'=
'=======================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var ds As DISCDLGSTRUCT
    Var Ret As Long

    ds.cbStructure = Len(ds)
    ds.hwndOwner = GethWnd
    ds.lpLocalName = StrAdr(Text1.GetWindowText & Chr$(0))
    ds.dwFlags = DISC_NO_FORCE Or DISC_UPDATE_PROFILE

    Ret = Api_WNetDisconnectDialog1(ds)

    If Ret = WN_SUCCESS Then
        Text1.SetWindowText ""
        A% = MessageBox("", "UnMapDrive Succeeded.", 0, 2)
    Else
        A% = MessageBox("", "Error!", 0, 2)
    End If
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

ネットワークプリンタ接続ダイアログボックスを表示

ConnectToPrinterDlg ネットワーク上のプリンタに接続するためのダイアログボックスを表示
ShellExecute 拡張子に関連付けられたプログラムを実行





「Open ExPlorer」をクリックすると下記ダイアログが表示されます。

```
'=====================================
'= ネットワークプリンタ接続ダイアログボックスを表示
'=    (ConnectToPrinterDlg.bas)
'=====================================
#include "Windows.bi"

#define SW_SHOWNORMAL 1                         'SW_RESTOREと同じ

' ネットワーク上のプリンタに接続するためのダイアログボックスを表示
Declare Function Api_ConnectToPrinterDlg& Lib "wi
nspool.drv" Alias "ConnectToPrinterDlg" (ByVal hWnd&, ByVal fla
gs&)

' 拡張子に関連付けられたプログラムを実行
Declare Function Api_ShellExecute& Lib "Shell32" Alias "ShellExecuteA" (ByVal hWnd&,
ByVal lpOperation$, ByVal lpFile$, ByVal lpParameters$, ByVal lpDirectory$, ByVal
nShowCmd&)

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    Ret = Api_ConnectToPrinterDlg(GethWnd, 0)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Ret As Long

    Ret = Api_ShellExecute(0, "Open", "explorer.exe",
"/e,::{2227A280-3AEA-1069-A2DE-08002B30309D}", ByVal 0, SW_SHOWNORMAL)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

「Connect Net Printer」をクリックすると、左記ダイアログが表示されます。

ネットワークプロバイダ名を取得

NetTypeで指定したタイプのネットワークプロバイダ名を取得します。
WNetGetProviderName ネットワークプロバイダ名を取得

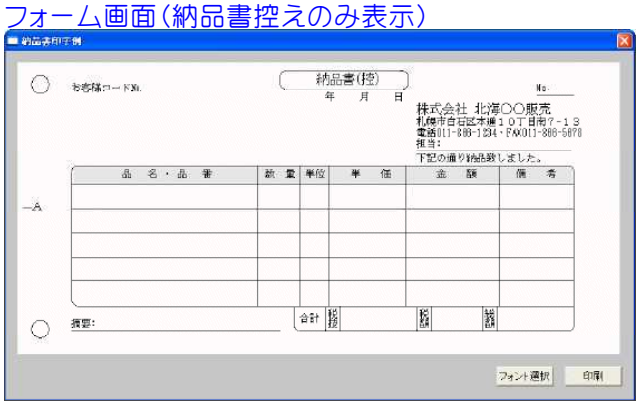

```
'=====================================
'= ネットワークプロバイダ名を取得
'=    (WNetGetProviderName.bas)
'=====================================
#include "Windows.bi"

' 指定した種類のネットワークのプロバイダ名を取得
Declare Function Api_WNetGetProviderName& Lib "mpr" Alias "WNetGetProviderNameA" (ByVal
dwNetType&, ByVal lpProvider$, lpBufferSize&)

#define WNNC_NET_LANMAN &H20000            'Microsoft Windows Network標準タイプ
#define ERROR_MORE_DATA 234                '
#define NO_ERROR 0                         '

Var Shared Text1 As Object
Var Shared Text2 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Text2.Attach GetDlgItem("Text2") : Text2.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long
    Var Provider As String
    Var BufferSize As Long

    'バッファサイズを取得
    Ret = Api_WNetGetProviderName(WNNC_NET_LANMAN, Provider, BufferSize)

    If Ret = ERROR_MORE_DATA Then

        '文字列変数を初期化
        Provider = String$(BufferSize + 1, Chr$(0))

        'ネットワークプロバイダ名を取得
        Ret = Api_WNetGetProviderName(WNNC_NET_LANMAN, Provider, BufferSize)
    End If

    If Ret = NO_ERROR Then

        'ネットワークプロバイダ名を表示
        Text2.SetWindowText Provider
    Else
        Text2.SetWindowText "ネットワークプロバイダ名を取得できません。"
    End If
End Sub

'=====================================
'=
'=====================================
```

```
While 1
    WaitEvent
Wend
Stop
End
```

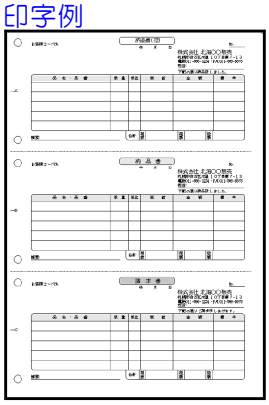
納品書印刷の一例

A4用紙縦置きで納品書(控)・納品書・請求書を印刷します。
3枚の伝票の境界には点線を入れます。
フォント・文字色(必要はありませんがテスト用)の設定ができます。フォントサイズはプログラム上で指定していますので無効です。

フォーム画面(納品書控えのみ表示)



印字例



```
'=======================================
'= 納品書印刷の一例
'=======================================
#include "Windows.bi"

Var Shared MAINFORM As Object
Var Shared PIC As Object
Var Shared PRT As Object

PRINTERObject PRT
Var Shared PRM As PRINTPARAM
Var Shared FFONT As FONT                    'フォント構造体
Var Shared F_NAME$ As String
Var Shared RGBCOLOR As Long

MAINFORM.ATTACH GETHWND
PIC.ATTACH GETDLGITEM("PICTURE1")

Declare Sub TEST_DRAW edecl ()

'=======================================
'=
'=======================================
Declare Sub MAINFORM_START edecl ()
Sub MAINFORM_START()
    SETWINDOWTEXT "納品書印字例"
    line(16,16)-(860,422),,15,bf
    F_NAME$ = "MS ゴシック"
    RGBCOLOR = RGB(0,0,0)
    PIC.SETFORECOLOR RGBCOLOR
    PIC.SETFONTNAME F_NAME$
    PIC.SETMAPMODE 2
    PIC.TEST_DRAW
End Sub

'=======================================
'=
'=======================================
Sub TEST_DRAW()
    Var LX As Integer                       'X軸オフセット
```

```
Var LY As Integer                       'Y軸オフセット
Var ZL As Integer                       '枠(Left)
Var ZU As Integer                       '枠(Top)
Var ZR As Integer                       '枠(Right)
Var ZB As Integer                       '枠(Bottom)
Var ZC As Integer                       '行間

SETFORECOLOR RGBCOLOR
ZA$(1) = "A"
ZA$(2) = "B"
ZA$(3) = "C"
ZB$(1) = "納品書(控)"
ZB$(2) = "納　品　書"
ZB$(3) = "請　求　書"
ZC$(1) = "下記の通り納品致しました。"
ZC$(2) = "下記の通り納品致しました。"
ZC$(3) = "下記の通りご請求申しあげます。"

'------------------------------
'納品書(控)･納品書･請求書
'------------------------------
For CT = 1 To 3
    goSub *PRT_STYLE
Next
Exit Sub

'------------------------------
*PRT_STYLE
'------------------------------
*PRT_LOC
If CT = 1 Then
    LY = 0
Else If CT = 2 Then
    LY = 990
Else
    LY = 1980
End If

ZL = 900 : ZU = 0 : ZR = 1350 : ZB = 60 : ZC = 20
If CT = 1 Then
    SETDRAWWIDTH 2
    goto *JP
Else If CT = 2 Then
    SETFILLCOLOR &HE0E0E0
    SETDRAWWIDTH 2
Else If CT = 3 Then
    SETFILLCOLOR &HC0C0C0
    SETDRAWWIDTH 3
End If
line(ZL+5,ZU+5+LY)-(ZR-5,ZB-5+LY),preset,,bf

*JP
SETFORECOLOR &H0
line  (ZL+ZC,ZU   +LY)-(ZR-ZC,ZU+LY)          ' ──
circle(ZL+ZC,ZU+ZC+LY),ZC,,,.5,.75            '┌
circle(ZR-ZC,ZU+ZC+LY),ZC,,,.75,0              '    ┐
line  (ZL   ,ZU+ZC+LY)-(ZL,ZB-ZC+LY)           '|
line  (ZR   ,ZU+ZC+LY)-(ZR,ZB-ZC+LY)           '    |
circle(ZL+ZC,ZB-ZC+LY),ZC,,,.25,.5             '└
circle(ZR-ZC,ZB-ZC+LY),ZC,,,0,.25              '    ┘
line  (ZL+ZC,ZB   +LY)-(ZR-ZC,ZB+LY)           ' ──
SETDRAWWIDTH 0

SETFILLCOLOR &HE0E0E0
line( 175, 315+LY)-(1925, 365+LY),preset,,bf

'------------------------------
'外枠角丸
'------------------------------
ZL = 170 : ZU = 310 : ZR = 1930 : ZB = 770
```

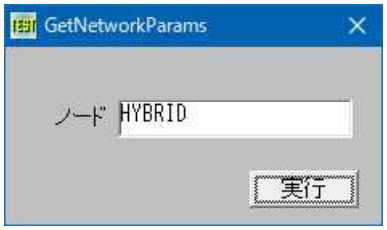
```
SETDRAWWIDTH 3
line  (ZL+ZC,ZU   +LY)-(ZR-ZC,ZU+LY)          ' ──
circle(ZL+ZC,ZU+ZC+LY),ZC,,,.5,.75            ' ┌
circle(ZR-ZC,ZU+ZC+LY),ZC,,,.75,0             '     ┐
line  (ZL   ,ZU+ZC+LY)-(ZL,ZB-ZC+LY)          ' │
circle(ZL+ZC,ZB-ZC+LY),ZC,,,.25,.5            ' └
line  (ZL+ZC,ZB   +LY)-(ZR,ZB+LY)             ' ──
ZB = 850 : ZR = 1930
line  (ZR   ,ZU+ZC+LY)-(ZR,ZB-ZC+LY)          '     ┘
circle(ZR-ZC,ZB-ZC+LY),ZC,,,0,.25
ZL = 950
line  (ZL+ZC,ZB   +LY)-(ZR-ZC,ZB+LY)          ' ──
circle(ZL+ZC,ZB-ZC+LY),ZC,,,.25,.5            ' └
ZU = 770
line  (ZL   ,ZU+LY)-(ZL,ZB-ZC+LY)             ' │
SETDRAWWIDTH 0

For I = 370 To 690 step 80
    line( 170,I+LY)-(ZR,I+LY)
Next

line( 170, 850+LY)-( 900, 850+LY)

line(1800,  80+LY)-(1930,  80+LY)
line(1380, 260+LY)-(1920, 260+LY)
line( 820, 310+LY)-( 820, 770+LY)
line( 970, 310+LY)-( 970, 770+LY)
line(1070, 310+LY)-(1070, 770+LY)

line(1340, 310+LY)-(1340, 770+LY)
line(1670, 310+LY)-(1670, 770+LY)

line(1060, 770+LY)-(1060, 850+LY)
line(1110, 770+LY)-(1110, 850+LY)
line(1380, 770+LY)-(1380, 850+LY)
line(1430, 770+LY)-(1430, 850+LY)
line(1610, 770+LY)-(1610, 850+LY)
line(1660, 770+LY)-(1660, 850+LY)

SETDRAWSTYLE 4
line(   0, 940+LY)-(1980, 940+LY)             '..... 切取線
SETDRAWSTYLE 0
circle(  60,  45+LY),30                       '○
line(   0, 445+LY)-(  30, 445+LY)             '-   伝票センター
SETFONTNAME F_NAME$
SETFONTSIZE 11
symbol(  30, 425+LY),ZA$(CT),1,1
circle(  60, 845+LY),30                       '○
SETFORECOLOR RGBCOLOR
SETFONTSIZE 9
symbol( 170,  40+LY),"お客様コード№",1,1
symbol(1800,  40+LY),"No.",1,1
symbol(1060,  70+LY),"年",1,1
symbol(1180,  70+LY),"月",1,1
symbol(1300,  70+LY),"日",1,1
SETFONTSIZE 12
symbol(1380, 105+LY),"株式会社 北海○○販売",1,1
SETFONTSIZE 9
symbol(1380, 150+LY),"札幌市白石区本通10丁目南7-13",1,1
symbol(1380, 185+LY),"電話011-866-1234•FAX011-866-5678",1,1
symbol(1380, 220+LY),"担当:",1,1
symbol(1380, 270+LY),ZC$(CT),1,1
symbol( 170, 810+LY),"摘要:",1,1
SETFONTSIZE 12
symbol(1030,   8+LY),ZB$(CT),1,1
SETFONTSIZE 9
symbol( 350, 320+LY),"品　名・品　番",1,1
symbol( 850, 320+LY),"数　量",1,1
symbol( 990, 320+LY),"単位",1,1
symbol(1150, 320+LY),"単　　価",1,1
```

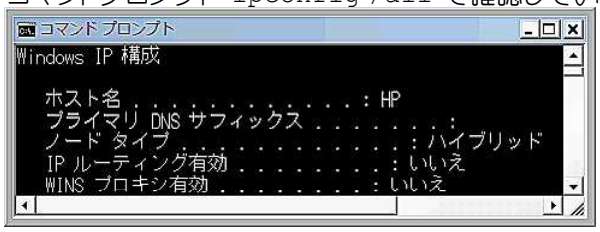
```
    symbol(1450, 320+LY),"金    額",1,1
    symbol(1730, 320+LY),"備    考",1,1
    symbol( 970, 795+LY),"合計",1,1
    symbol(1070, 780+LY),"税",1,1
    symbol(1070, 810+LY),"抜",1,1
    symbol(1390, 780+LY),"税",1,1
    symbol(1390, 810+LY),"額",1,1
    symbol(1620, 780+LY),"総",1,1
    symbol(1620, 810+LY),"額",1,1
    return
End Sub

'=======================================
'=  フォント選択(フォントスタイル･サイズは無効:プログラム上で指定)
'=======================================
Declare Sub BUTTON1_ON edecl ()
Sub BUTTON1_ON()
    FFONT.FFNAME = F_NAME$
    If CHOOSEFONT(FFONT, RGBCOLOR, -1, 0) Then
        F_NAME$ = kLeft$(FFONT.FFNAME,kinstr(FFONT.FFNAME, Chr$(0))-1)
    End If

    PIC.cls
    PIC.TEST_DRAW
End Sub

'=======================================
'=  印刷ダイアログ
'=======================================
Declare Sub BUTTON2_ON edecl ()
Sub BUTTON2_ON()
    PRT.SETUPPRINTERMODE "SETUPPRINTER:",9,1
    If PRT.PRINTDLG(PRM) = 0 Then Exit Sub
    PRT.SETMAPMODE 2
    PRT.STARTDOC "Sample"
    PRT.STARTPAGE

    PRT.TEST_DRAW

    PRT.ENDPAGE
    PRT.ENDDOC
    PRT.CLOSEPRINTER
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

ノードタイプを取得

GetNetworkParams ネットワーク構成値を取得
MoveMemory メモリの指定領域をコピー

コマンドプロンプト ipconfig /all で確認しています。





名称	動作
ブロードキャスト	IPのブロードキャストで名前解決を図る。
ピア ツー ピア	NetBIOS用のネームサーバを利用して名前解決を図る。
複合	最初にブロードキャストを試し、失敗したらネーム サーバを利用する。
ハイブリッド	最初にネーム サーバを利用し、失敗したらブロードキャストを試す。

```
'=====================================
'= ノードタイプを取得
'=    (GetNetworkParams.bas)
'=====================================
#include "Windows.bi"

Type IP_ADDRESS_STRING
    IpAddressString(4 * 4 - 1) As Byte
End Type

Type IP_MASK_STRING
    IpMaskString(4 * 4 - 1) As Byte
End Type

Type IP_ADDR_STRING
    iNext      As Long
    IpAddress  As IP_ADDRESS_STRING
    IpMask     As IP_MASK_STRING
    Context    As Long
End Type

#define MAX_HOSTNAME_LEN 128              'ホスト名の最大長
#define MAX_DOMAIN_NAME_LEN 128           'ドメイン名の最大長
#define MAX_SCOPE_ID_LEN 256              'スコープIDの最大長

#define ERROR_NOT_SUPPORTED 50            'ネットワーク要求非サポートを示す
#define ERROR_BUFFER_OVERFLOW 111         'バッファサイズ不足を示す
#define ERROR_INVALID_PARAMETER 87        'パラメータ不正を示す
#define ERROR_NO_DATA 232                 'アダプタ情報不在を示す定数の宣言

#define BROADCAST_NODETYPE 1              'ブロードキャスト(b-node)
#define PEER_TO_PEER_NODETYPE 2           'ピアツーピア(p-node)
#define MIXED_NODETYPE 4                  '混合(m-node)
#define HYBRID_NODETYPE 8                 'ハイブリッド(h-node)

Type FIXED_INFO
    HostName(MAX_HOSTNAME_LEN + 4 - 1)      As Byte
    DomainName(MAX_DOMAIN_NAME_LEN + 4 - 1) As Byte
    CurrentDnsServer                        As Long
    DnsServerList                           As IP_ADDR_STRING
    NodeType                                As Long
    ScopeId(MAX_SCOPE_ID_LEN + 4 - 1)       As Byte
    EnableRouting                           As Long
    EnableProxy                             As Long
    EnableDns                               As Long
End Type

' ネットワーク構成値を取得
Declare Function Api_GetNetworkParams& Lib "iphlpapi" Alias "GetNetworkParams" (pfi As
Any, pOutBufLen&)

' メモリの指定領域をコピー
Declare Sub MoveMemory Lib "Kernel32" Alias "RtlMoveMemory" (Dest As Any, Source As Any,
ByVal length&)

Var Shared Text1 As Object
Var Shared Text2 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Text2.Attach GetDlgItem("Text2") : Text2.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
```

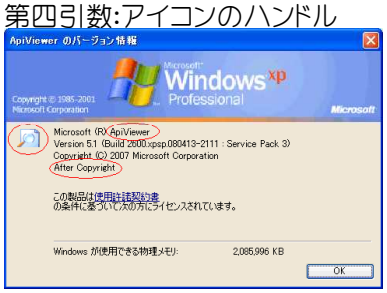
```
'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Needed As Long
    Var fi As FIXED_INFO
    Var ScopeId As String
    Var Ret As Long

    'バッファ必要サイズを取得
    Ret = Api_GetNetworkParams(ByVal 0, Needed)

    'バッファサイズ不足のとき
    If Ret = ERROR_BUFFER_OVERFLOW Then

        'バッファを確保
        Var Buffer(Needed) As Byte

    'その他のとき
    Else

        'エラー情報を表示(戻り値によって分岐)
        Select Case Ret

            'パラメータ不正
            Case ERROR_INVALID_PARAMETER
                Text2.SetWindowText "パラメータが不正です。"

            'アダプタ情報不在
            Case ERROR_NO_DATA
                Text2.SetWindowText "アダプタ情報が存在しません。"

            'ネットワーク要求非サポート
            Case ERROR_NOT_SUPPORTED
                Text2.SetWindowText "ネットワーク要求は" & "サポートされていません。"

            'その他
            Case Else
                Text2.SetWindowText "(" & Str$(Ret) & ")" & "アダプタ情報は取得できません。"
        End Select

        '取得できない
        Exit Sub
    End If

    'ネットワーク構成値を取得
    Ret = Api_GetNetworkParams(Buffer(0), Needed)

    'ネットワーク構成値を構造体へ移動
    MoveMemory fi, Buffer(0), Len(fi)

    'ネットワーク構成値を表示(ノードタイプによって分岐)
    Select Case fi.NodeType
        Case BROADCAST_NODETYPE             'ブロードキャスト
            Text2.SetWindowText "BROADCAST"
        Case PEER_TO_PEER_NODETYPE          'ピアツーピア
            Text2.SetWindowText "PEER_TO_PEER"
        Case MIXED_NODETYPE                 'ミックス
            Text2.SetWindowText "MIXED"
        Case HYBRID_NODETYPE                'ハイブリッド
            Text2.SetWindowText "HYBRID"
        Case Else
            Text2.SetWindowText "UNKNOWN"   'その他
    End Select
End Sub

'=====================================
'=
'=====================================
```

```
While 1
    WaitEvent
Wend
Stop
End
```

バージョン情報ダイアログボックスを表示

ShellAbout 標準の「バージョン情報」ダイアログボックスを表示
ExtractIcon EXE･DLLからアイコンを取得

第一引数:ウィンドウハンドル
第二引数:タイトル
第三引数:Copyright (C) 1981-2001 Microsoft Corporation の後に続く文字列
第四引数:アイコンのハンドル

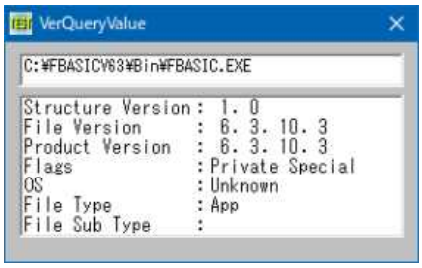


メモ帳のバージョン情報の例

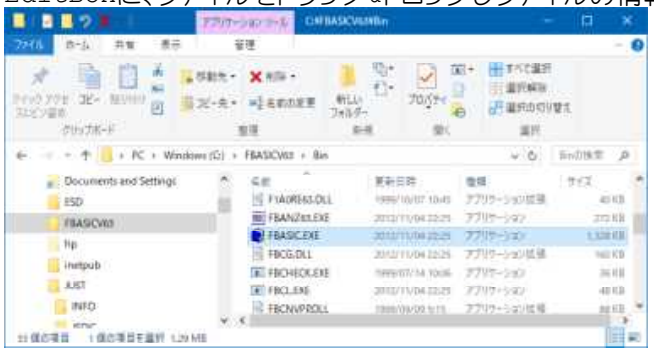
```
'=====================================
'= バージョン情報ダイアログボックスを表示
'=    (ShellAbout.bas)
'=====================================

' EXE･DLLからアイコンを取得
Declare Function Api_ExtractIcon& Lib "shell32" Alias "ExtractIconA" (ByVal hInst&,
ByVal Path$, ByVal Index&)

' バージョン情報ダイアログボックスを表示
Declare Function Api_ShellAbout& Lib "shell32" Alias "ShellAboutA" (ByVal hWnd&, ByVal
szApp$, ByVal szOtherStuff$, ByVal hIcon&)

Var hIcon As Long
Var Ret As Long
hIcon = Api_ExtractIcon(GethWnd, "C:¥_FB_API_E¥_ApiViewer¥ApiViewer.exe", 0)

Ret = Api_ShellAbout(GethWnd, "ApiViewer", "After Copyright", hIcon)

Stop
End
```

バージョン情報リソースから選択された情報を取得

GetFileVersionInfo ファイルに関するバージョン情報を取得
GetFileVersionInfoSize ファイル内のバージョン情報のサイズを取得
VerQueryValue バージョン情報リソースから選択されたバージョン情報を取得
GetSystemDirectory Windows のシステムディレクトリのパスを取得
MoveMemory メモリの指定領域をコピー



EditBoxに、ファイルをドラッグ&ドロップしファイルの情報を得ています。



```
'=====================================
'= バージョン情報リソースから選択された情報を取得
'=    (VerQueryValue.bas)
'=====================================
#include "Windows.bi"

' ファイルに関するバージョン情報を取得
Declare Function Api_GetFileVersionInfo& Lib "Version" Alias "GetFileVersionInfoA"
(ByVal lptstrFilename$, ByVal dwhandle&, ByVal dwlen&, lpData As Any)

' ファイル内のバージョン情報のサイズを取得
Declare Function Api_GetFileVersionInfoSize& Lib "Version" Alias
"GetFileVersionInfoSizeA" (ByVal lptstrFilename$, lpdwHandle&)

' バージョン情報リソースから選択されたバージョン情報を取得
Declare Function Api_VerQueryValue& Lib "Version" Alias "VerQueryValueA" (pBlock As Any,
ByVal lpSubBlock$, lplpBuffer As Any, puLen&)

' Windows のシステムディレクトリのパスを取得。システムディレクトリには、Windows ライブラリ、ドライバなどのファイルが置かれている
Declare Function Api_GetSystemDirectory& Lib "Kernel32" Alias "GetSystemDirectoryA"
(ByVal lpBuffer$, ByVal nSize&)

' メモリの指定領域をコピー
Declare Sub MoveMemory Lib "Kernel32" Alias "RtlMoveMemory" (Dest As Any, ByVal Source&,
ByVal Length&)

Type VS_FIXEDFILEINFO
    dwSignature             As Long
    dwStrucVersionl         As Integer
    dwStrucVersionh         As Integer
    dwFileVersionMSl        As Integer
    dwFileVersionMSh        As Integer
    dwFileVersionLSl        As Integer
    dwFileVersionLSh        As Integer
    dwProductVersionMSl     As Integer
    dwProductVersionMSh     As Integer
    dwProductVersionLSl     As Integer
    dwProductVersionLSh     As Integer
    dwFileFlagsMask         As Long
    dwFileFlags             As Long
    dwFileOS                As Long
    dwFileType              As Long
    dwFileSubtype           As Long
    dwFileDateMS            As Long
    dwFileDateLS            As Long
End Type

#define VS_FFI_FILEFLAGSMASK &H3F          '
#define VS_FFI_SIGNATURE &HFEEF04BD        '構造体の識別子
#define VS_FFI_STRUCVERSION &H10000        '構造体のバイナリバージョン番号0x00000029以上の値

#define VS_FF_DEBUG &H1                    'デバッグ情報が含まれている
#define VS_FF_INFOINFERRED &H10            '構造体は動的に作成されている
#define VS_FF_PATCHED &H4                  'ファイルは修正されたバージョン
#define VS_FF_PRERELEASE &H2               'ファイルは開発バージョン
```

```
#define VS_FF_PRIVATEBUILD &H8              '標準的なリリースされたファイルではない
#define VS_FF_SPECIALBUILD &H20             '標準ファイルのバリエーション

#define VOS__BASE &H0                       '
#define VOS__PM16 &H2                       '
#define VOS__PM32 &H3                       '
#define VOS__WINDOWS16 &H1                  '
#define VOS__WINDOWS32 &H4                  '
#define VOS_DOS &H10000                     'MS-DOS
#define VOS_DOS_WINDOWS16 &H10001           'Win16
#define VOS_DOS_WINDOWS32 &H10004           'Win32s
#define VOS_NT &H40000                      'WindowsNT
#define VOS_NT_WINDOWS32 &H40004            '設計対象となったOS
#define VOS_OS216 &H20000                   'OS/2  16ビット
#define VOS_OS216_PM16 &H20002              '
#define VOS_OS232 &H30000                   'OS/2  32ビット
#define VOS_OS232_PM32 &H30003              '
#define VOS_UNKNOWN &H0                     '

#define VFT_APP &H1                         'アプリケーション
#define VFT_DLL &H2                         'ダイナミックリンクライブラリ
#define VFT_DRV &H3                         'デバイスドライバ
#define VFT_FONT &H4                        'フォント
#define VFT_STATIC_LIB &H7                  'スタティックリンクライブラリ
#define VFT_UNKNOWN &H0                     '
#define VFT_VXD &H5                         '仮想デバイス

#define VFT2_DRV_COMM &HA                   '
#define VFT2_DRV_DISPLAY &H4                'ディスプレイドライバ
#define VFT2_DRV_INSTALLABLE &H8            'インストール可能なドライバ
#define VFT2_DRV_KEYBOARD &H2               'キーボードドライバ
#define VFT2_DRV_LANGUAGE &H3               '言語ドライバ
#define VFT2_DRV_MOUSE &H5                  'マウスドライバ
#define VFT2_DRV_NETWORK &H6                'ネットワークドライバ
#define VFT2_DRV_PRINTER &H1                'プリンタドライバ
#define VFT2_DRV_SOUND &H9                  'サウンドドライバ
#define VFT2_DRV_SYSTEM &H7                 'ファイルはシステムドライバ
#define VFT2_UNKNOWN &H0                    '

#define VFT2_FONT_RASTER &H1                'ラスタフォント
#define VFT2_FONT_VECTOR &H2                'ベクタフォント
#define VFT2_FONT_TRUETYPE &H3              'TrueTypeフォント

#define MAX_PATH 260

Var Shared Edit1 As Object
Var Shared List1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 12
List1.Attach GetDlgItem("List1") : List1.SetFontSize 14 : List1.SetWindowSize 274, 104

Var Shared sPath As String

'=====================================
'=
'=====================================
Declare Sub DisplayVerInfo(FilePath As String)
Sub DisplayVerInfo(FilePath As String)
    Var SVer As String
    Var FVer As String
    Var PVer As String
    Var F As String
    Var Typ As String
    Var STyp As String
    Var OS As String
    Var l As Long
    Var BuffL As Long
    Var Pointer As Long
    Var VERSION As VS_FIXEDFILEINFO
    Var Ret As LOng
```

```
    List1.ResetContent

    l = Api_GetFileVersionInfoSize(FilePath, 0)
    If l < 1 Then
        A% = MessageBox("", "情報を取得できません！", 0, 2)
    Else
        Var Buff(l) As Byte
        Ret = Api_GetFileVersionInfo(FilePath, 0, l, Buff(0))
        Ret = Api_VerQueryValue(Buff(0), "¥", Pointer, BuffL)
        MoveMemory VERSION, Pointer, Len(VERSION)

        SVer = Format$(VERSION.dwStrucVersionh) & "." & Format$(VERSION.dwStrucVersionl)

        FVer = Format$(VERSION.dwFileVersionMSh) & "." &
Format$(VERSION.dwFileVersionMSl) & "." & Format$(VERSION.dwFileVersionLSh) & "." &
Format$(VERSION.dwFileVersionLSl)

        PVer = Format$(VERSION.dwProductVersionMSh) & "." &
Format$(VERSION.dwProductVersionMSl) & "." & Format$(VERSION.dwProductVersionLSh) & "."
& Format$(VERSION.dwProductVersionLSl)

        If VERSION.dwFileFlags And VS_FF_DEBUG Then F = "Debug "
        If VERSION.dwFileFlags And VS_FF_PRERELEASE Then F = F & "PreRel "
        If VERSION.dwFileFlags And VS_FF_PATCHED Then F = F & "Patched "
        If VERSION.dwFileFlags And VS_FF_PRIVATEBUILD Then F = F & "Private "
        If VERSION.dwFileFlags And VS_FF_INFOINFERRED Then F = F & "Info "
        If VERSION.dwFileFlags And VS_FF_SPECIALBUILD Then F = F & "Special "
        If VERSION.dwFileFlags And VFT2_UNKNOWN Then F = F & "Unknown "

        Select Case VERSION.dwFileOS
            Case VOS_DOS_WINDOWS16
                OS = "DOS-Win16"
            Case VOS_DOS_WINDOWS32
                OS = "DOS-Win32"
            Case VOS_OS216_PM16
                OS = "OS/2-16 PM-16"
            Case VOS_OS232_PM32
                OS = "OS/2-16 PM-32"
            Case VOS_NT_WINDOWS32
                OS = "NT-Win32"
            Case Else
                OS = "Unknown"
        End Select

        Select Case VERSION.dwFileType
            Case VFT_APP
                Typ = "App"
            Case VFT_DLL
                Typ = "DLL"
            Case VFT_DRV
                Typ = "Driver"
                Select Case VERSION.dwFileSubtype
                    Case VFT2_DRV_PRINTER
                        STyp = "Printer drv"
                    Case VFT2_DRV_KEYBOARD
                        STyp = "Keyboard drv"
                    Case VFT2_DRV_LANGUAGE
                        STyp = "Language drv"
                    Case VFT2_DRV_DISPLAY
                        STyp = "Display drv"
                    Case VFT2_DRV_MOUSE
                        STyp = "Mouse drv"
                    Case VFT2_DRV_NETWORK
                        STyp = "Network drv"
                    Case VFT2_DRV_SYSTEM
                        STyp = "System drv"
                    Case VFT2_DRV_INSTALLABLE
                        STyp = "Installable"
                    Case VFT2_DRV_SOUND
                        STyp = "Sound drv"
```

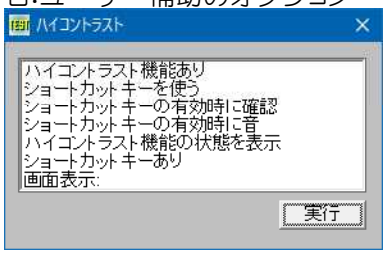
```
                    Case VFT2_DRV_COMM
                        STyp = "Comm drv"
                    Case VFT2_UNKNOWN
                        STyp = "Unknown"
                End Select
            Case VFT_FONT
                Typ = "Font"
                Select Case VERSION.dwFileSubtype
                    Case VFT2_FONT_RASTER
                        STyp = "Raster Font"
                    Case VFT2_FONT_VECTOR
                        STyp = "Vector Font"
                    Case VFT2_FONT_TRUETYPE
                        STyp = "TrueType Font"
                End Select
            Case VFT_VXD
                Typ = "VxD"
            Case VFT_STATIC_LIB
                Typ = "Lib"
            Case Else
                Typ = "Unknown"
        End Select
    End If

    List1.AddString "Structure Version:" & SVer
    List1.AddString "File Version     :" & FVer
    List1.AddString "Product Version  :" & PVer
    List1.AddString "Flags           :" & F
    List1.AddString "OS              :" & OS
    List1.AddString "File Type        :" & Typ
    List1.AddString "File Sub Type    :" & STyp
End Sub

'=====================================
'=
'=====================================
Declare Sub Edit1_Change edecl ()
Sub Edit1_Change()
    Var Ret As Long

    DisplayVerInfo sPath
End Sub

'=====================================
'= シェルドロップされたファイル名を取得
'=====================================
Declare Sub Edit1_DropFiles edecl (ByVal DF As Long)
Sub Edit1_DropFiles(ByVal DF As Long)
    Var Ret As Long

    Ret = GetDropFileCount(DF)
    sPath = GetDropFileName(DF, 0)
    Edit1.SetWindowText sPath
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

ハイコントラスト機能情報を取得

ハイコントラスト機能の情報を取得します。

SystemParametersInfo システム全体に関するパラメータを取得・設定
lstrcpy 文字列をコピーする

右:ユーザー補助のオプション → 画面 → ハイコントラストの設定 で一部を確認しています。

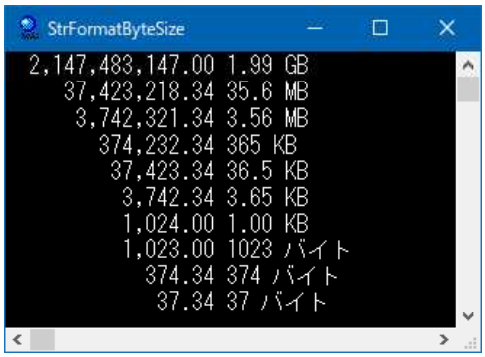



```
'=====================================
'= ハイコントラスト機能情報を取得
'=    (SPI_GETHIGHCONTRAST.bas)
'=====================================
#include "Windows.bi"

Type HIGHCONTRAST
    cbSize            As Long
    dwFlags           As Long
    lpszDefaultScheme As Long
End Type

#define HCF_AVAILABLE &H2                 'ハイコントラストモード機能あり
#define HCF_CONFIRMHOTKEY &H8             'ショートカットキーの有効時に確認
#define HCF_HIGHCONTRASTON &H1            'ハイコントラスト機能有効
#define HCF_HOTKEYACTIVE &H4              '[Alt]+[Shift]+[PrintScreen]でショートカットキーを使う

#define HCF_HOTKEYAVAILABLE &H40          'ショートカットキーあり
#define HCF_HOTKEYSOUND &H10              'ショートカットキーの有効時に音
#define HCF_INDICATOR &H20                'ハイコントラスト機能の状態を表示
#define SPI_GETHIGHCONTRAST 66            'ユーザー補助機能のハイコントラストの設定状況を定義するHIGHCONTRAST構造体を取得

' システム全体に関するパラメータを取得・設定
Declare Function Api_SystemParametersInfo& Lib "user32" Alias "SystemParametersInfoA"
(ByVal uiAction&, ByVal uiParam&, pvParam As Any, ByVal fWinIni&)

' 文字列をコピーする
Declare Function Api_lstrcpy& Lib "Kernel32" Alias "lstrcpyA" (ByVal lpszString1$,
lpszString2 As Any)

Var Shared List1 As Object
Var Shared Button1 As Object

List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hc As HIGHCONTRAST
    Var Scheme As String * 255
    Var Ret As Long

    'リストボックスをクリア
    List1.Resetcontent

    '構造体を初期化
    hc.cbSize = Len(hc)

    'ハイコントラスト機能の情報を取得
    Ret = Api_SystemParametersInfo(SPI_GETHIGHCONTRAST, Len(hc), hc, 0)

    'ハイコントラスト機能の情報を表示
    If (hc.dwFlags And HCF_HIGHCONTRASTON) = HCF_HIGHCONTRASTON Then
```

```
        List1.AddString "ハイコントラスト機能有効"
    End If

    If (hc.dwFlags And HCF_AVAILABLE) = HCF_AVAILABLE Then
        List1.AddString "ハイコントラスト機能あり"
    End If

    If (hc.dwFlags And HCF_HOTKEYACTIVE) = HCF_HOTKEYACTIVE Then
        List1.AddString "ショートカットキーを使う"
    End If

    If (hc.dwFlags And HCF_CONFIRMHOTKEY) = HCF_CONFIRMHOTKEY Then
        List1.AddString "ショートカットキーの有効時に確認"
    End If

    If (hc.dwFlags And HCF_HOTKEYSOUND) = HCF_HOTKEYSOUND Then
        List1.AddString "ショートカットキーの有効時に音"
    End If

    If (hc.dwFlags And HCF_INDICATOR) = HCF_INDICATOR Then
        List1.AddString "ハイコントラスト機能の状態を表示"
    End If

    If (hc.dwFlags And HCF_HOTKEYAVAILABLE) = HCF_HOTKEYAVAILABLE Then
        List1.AddString "ショートカットキーあり"
    End If

    Ret = Api_lstrcpy(Scheme, ByVal hc.lpszDefaultScheme)
    List1.AddString "表示設定:" & Left$(Scheme, InStr(Scheme, Chr$(0)) - 1)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

バイトサイズを表す数値を文字列に変換

StrFormatByteSize バイトサイズを表す数値を文字列に変換します。

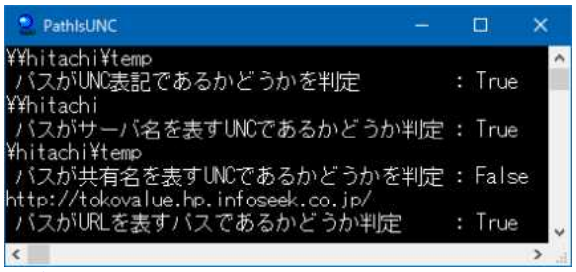



```
'=====================================
'= バイトサイズを表す数値を文字列に変換
'=    (StrFormatByteSize.bas)
'=====================================

' バイトサイズを表す数値を文字列に変換
Declare Function Api_StrFormatByteSize& Lib "shlwapi" Alias "StrFormatByteSizeA" (ByVal
dw&, ByVal pszBuf$, ByVal cchBuf&)
```

```
'=====================================
'=
'=====================================
Declare Function FormatByteSize(dwBytes As Single) As String
Function FormatByteSize(dwBytes As Single) As String
    Var Buff As String
    Var dwBuff As Long

    Buff = Space$(32)
    dwBuff = Len(Buff)

    If Api_StrFormatByteSize(dwBytes, Buff, dwBuff) <> 0 Then
        FormatByteSize = Left$(Buff, InStr(Buff, Chr$(0)) - 1)
    End If
End Function

'------------------------------
Print format$(2147483147 , "###,###,###,###.## ") & FormatByteSize(2147483147)
Print format$(37423218.34, "###,###,###,###.## ") & FormatByteSize(37423218.34)
Print format$(3742321.34 , "###,###,###,###.## ") & FormatByteSize(3742321.34)
Print format$(374232.34  , "###,###,###,###.## ") & FormatByteSize(374232.34)
Print format$(37423.34   , "###,###,###,###.## ") & FormatByteSize(37423.34)
Print format$(3742.34    , "###,###,###,###.## ") & FormatByteSize(3742.34)
Print format$(1024       , "###,###,###,###.## ") & FormatByteSize(1024)
Print format$(1023       , "###,###,###,###.## ") & FormatByteSize(1023)
Print format$(374.34     , "###,###,###,###.## ") & FormatByteSize(374.34)
Print format$(37.34      , "###,###,###,###.## ") & FormatByteSize(37.34)

Stop
End
```

パスがUNC関連であるかどうかの判定

PathIsUNC パスがUNC表記であるかどうかを判定
PathIsUNCServer パスがサーバ名を表すUNCであるかどうか判定
PathIsUNCServerShare パスが共有名を表すUNCであるかどうかを判定
PathIsURL パスがURLを表すパスであるかどうか判定



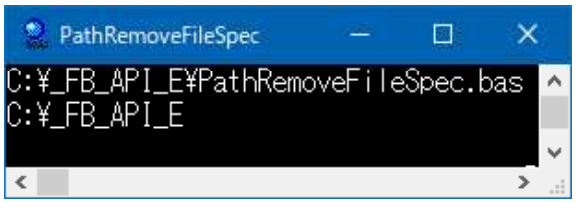
参考
UNC(Universal Naming Convention)とは、Microsoft社のWinodows向けネットワーク環境「Microsoftネットワーク」上で、ネットワークの上にあるマシン上の資源(ファイル、プリンタなど)を表示するための表記法。

```
'=====================================
'= パスがUNC関連であるかどうかの判定
'=    (PathIsUNC.bas/P)
'=====================================

' パスがUNC表記であるかどうかを判定
Declare Function Api_PathIsUNC& Lib "shlwapi" Alias "PathIsUNCA" (ByVal pszPath$)

' パスがサーバ名を表すUNCであるかどうか判定
Declare Function Api_PathIsUNCServer& Lib "shlwapi" Alias "PathIsUNCServerA" (ByVal
pszPath$)

' パスが共有名を表すUNCであるかどうかを判定
Declare Function Api_PathIsUNCServerShare& Lib "shlwapi" Alias "PathIsUNCServerShareA"
(ByVal pszPath$)
```

```
' パスがURLを表すパスであるかどうか判定
Declare Function Api_PathIsURL& Lib "shlwapi" Alias "PathIsURLA" (ByVal pszPath$)

Var PathName As String
Var txt As String
Var Ret As Long

PathName = "¥¥hitachi¥temp"
Ret = Api_PathIsUNC(PathName)
GoSub *Judge
Print " パスがUNC表記であるかどうかを判定          : " & txt

PathName = "¥¥hitachi"
Ret = Api_PathIsUNCServer(PathName)
GoSub *Judge
Print " パスがサーバ名を表すUNCであるかどうか判定 : " & txt

PathName = "¥hitachi¥temp"
Ret = Api_PathIsUNCServerShare(PathName)
GoSub *Judge
Print " パスが共有名を表すUNCであるかどうかを判定 : " & txt

PathName = "http://tokovalue.jp/"
Ret = Api_PathIsURL(PathName)
GoSub *Judge
Print " パスがURLを表すパスであるかどうか判定      : " & txt

Stop
End

*judge
    If Ret Then txt = "True" Else txt = "False"
    Print PathName
    Return
```

パスからファイル名部分を取り除く

PathRemoveFileSpec パスからファイル名部分を取り除く




```
'=====================================
'= パスからファイル名部分を取り除く
'=   (PathRemoveFileSpec.bas/P)
'=====================================

' パスからファイル名部分を取り除く
Declare Sub Api_PathRemoveFileSpec Lib "shlwapi" Alias "PathRemoveFileSpecA" (ByVal
pPath$)

Var Path As String

Path = "C:¥_FB_API_E¥PathRemoveFileSpec.bas"
Print Path

Api_PathRemoveFileSpec Path
Print Left$(Path, InStr(Path, Chr$(0)) - 1)

Stop
End
```

パスからルートパスを取り出す

PathStripToRoot パスからルートパスを取り出す

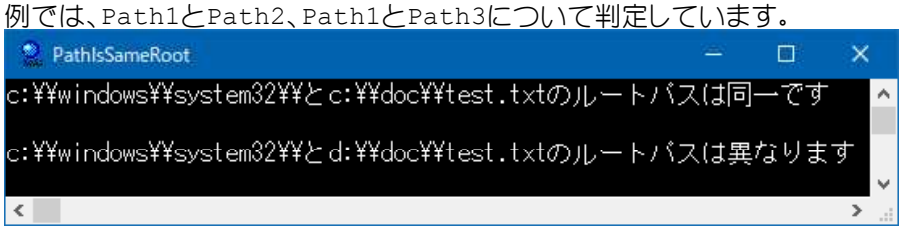

```
'=====================================
'= パスからルートパスを取り出す
'=    (PathStripToRoot.bas)
'=====================================
#include "Windows.bi"

' パスからルートパスを取り出す
Declare Function Api_PathStripToRoot& Lib "shlwapi" Alias "PathStripToRootA" (ByVal
pszPath$)
Var Shared Edit1 As Object
Var Shared Text1 As Object
Var Shared Button1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'= Chr$(0)を取り除く
'=====================================
Declare Function TrimNull (item As String) As String
Function TrimNull(item As String) As String
    Var ePos As Integer

    ePos = Instr(item, Chr$(0))
    If ePos Then
        TrimNull = Left$(item, ePos - 1)
    Else
        TrimNull = item
    End If
End Function

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Edit1.SetWindowText "C:¥THE DIR¥MYFILE¥"
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var sSave As String
    Var Ret As Long

    sSave = Edit1.GetWindowText & String$(100, 0)

    Ret = Api_PathStripToRoot(sSave)

    Text1.SetWindowText TrimNull(sSave)
End Sub
```

```
'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

パス同士が同一ルートパスかどうかの判定

PathIsSameRoot 二つのパスが同一のルートパスを持つかどうか判定

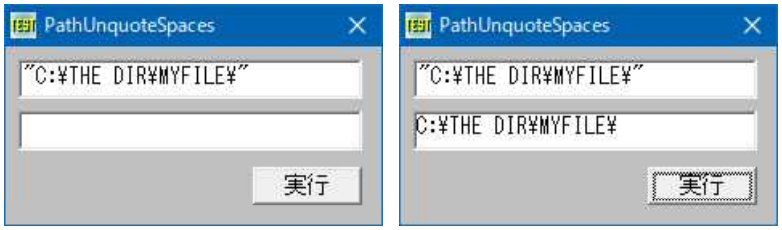
例では、Path1とPath2、Path1とPath3について判定しています。




```
'=====================================
'= 二つのパスが同一のルートパスを持つかどうか判定
'=    (PathIsSameRoot.bas/P)
'=====================================

' 二つのパスが同一のルートパスを持つかどうか判定
Declare Function Api_PathIsSameRoot& Lib "shlwapi" Alias "PathIsSameRootA" (ByVal
pszPath1$, ByVal pszPath2$)

Var Path1 As String
Var Path2 As String
Var Path3 As String
Path1 = "c:¥¥windows¥¥system32¥¥"
Path2 = "c:¥¥doc¥¥test.txt"
Path3 = "d:¥¥doc¥¥test.txt"

If (Api_PathIsSameRoot(Path1, Path2)) Then
    Print Path1 & "と" & Path2 & "のルートパスは同一です"
Else
    Print Path1 & "と" & Path2 & "のルートパスは異なります"
End If

Print

If (Api_PathIsSameRoot(Path1, Path3)) Then
    Print Path1 & "と" & Path3 & "のルートパスは同一です"
Else
    Print Path1 & "と" & Path3 & "のルートパスは異なります"
End If

Stop
End
```

パスの前後のダブルクォーテーションを取り除く

PathUnquoteSpaces パスの前後のダブルクォーテーションを取り除く

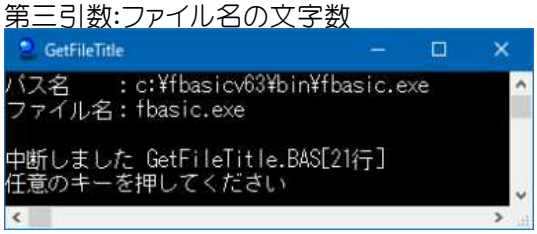



```
'=====================================
'= パスの前後のダブルクォーテーションを取り除く
'=    (PathUnquoteSpaces.bas)
'=====================================
#include "Windows.bi"

' パスの前後のダブルクォーテーションを取り除く
Declare Sub Api_PathUnquoteSpaces Lib "shlwapi" Alias "PathUnquoteSpacesA" (ByVal lpsz$)

Var Shared Edit1 As Object
Var Shared Text1 As Object
Var Shared Button1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'= Chr$(0)を取り除く
'=====================================
Declare Function TrimNull (item As String) As String
Function TrimNull(item As String) As String
    Var ePos As Integer

    ePos = Instr(item, Chr$(0))
    If ePos Then
        TrimNull = Left$(item, ePos - 1)
    Else
        TrimNull = item
    End If
End Function

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Edit1.SetWindowText Chr$(&H22) & "C:¥THE DIR¥MYFILE¥" & Chr$(&H22)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var sSave As String

    sSave = Edit1.GetWindowText & String$(100, 0)

    Api_PathUnquoteSpaces sSave

    Text1.SetWindowText TrimNull(sSave)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop : End
```

パス名からファイル名を取得(Ⅰ)

GetFileTitle パス名からファイル名を取得

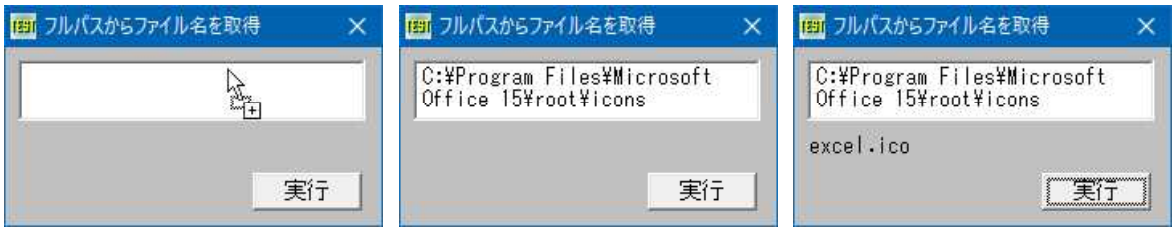
第一引数:ファイルのパスを指定
第二引数:ファイル名
第三引数:ファイル名の文字数




```
'=====================================
'= パスからファイル名を取得
'=    (GetFileTitle.bas)
'=====================================
#include "Windows.bi"

' パスからファイル名を取得
Declare Function Api_GetFileTitle% Lib "comdlg32" Alias "GetFileTitleA" (ByVal
lpszFile$, ByVal lpszTitle$, ByVal cbBuf%)
Var TestString As String
Var Buffer As String
Var Ret As Integer

Buffer = string$(255, 0)
TestString = "c:¥fbasicv63¥bin¥fbasic.exe"

Ret = Api_GetFileTitle(TestString, Buffer, len(Buffer))
Buffer = left$(Buffer, instr(1, Buffer, Chr$(0)) - 1)

Print "パス名 :" & TestString
Print "ファイル名:" & Buffer
Stop
End
```

パス名からファイル名を取得(Ⅱ)

フルパス名からファイル名を抜き出します。
例では、EditBoxにファイルをドラッグ＆ドロップしフルパス名を表示させ、「実行」ボタンクリックで抜き出したファイル名のみを表示させています。
FindFirstFile ファイルやディレクトリを検索
FindClose ファイル検索ハンドルを解放




```
'=====================================
'= フルパス名からファイル名を取得
'=    (FindFirstFile3.bas)
'=====================================
#include "Windows.bi"

#define FILE_ATTRIBUTE_DIRECTORY &H10      'ディレクトリ属性
#define INVALID_HAndLE_VALUE -1            '見つからない場合
#define MAX_PATH 260
```

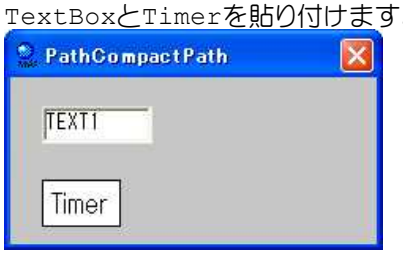
```
Type FILETIME
    dwLowDateTime     As Long
    dwHighDateTime    As Long
End Type

Type WIN32_FIND_DATA
    dwFileAttributes  As Long
    ftCreationTime    As FILETIME
    ftLastAccessTime  As FILETIME
    ftLastWriteTime   As FILETIME
    nFileSizeHigh     As Long
    nFileSizeLow      As Long
    dwReserved0       As Long
    dwReserved1       As Long
    cFileName         As String * MAX_PATH
    cAlternate        As String * 14
End Type

' 指定したファイル名に一致するファイルやディレクトリを検索
Declare Function Api_FindFirstFile& Lib "Kernel32" Alias "FindFirstFileA" (ByVal
lpFileName$, lpFindFileData As WIN32_FIND_DATA)

' ファイル検索ハンドルをクローズ
Declare Function Api_FindClose& Lib "Kernel32" Alias "FindClose" (ByVal hFindFile&)
Var Shared Edit1 As Object
Var Shared Text1 As Object
Var Shared Button1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontsize 14
Var Shared FullPath As String

'=====================================
'= Chr$(0)を除去
'=====================================
Declare Function TrimNull(StartStr As String) As String
Function TrimNull(StartStr As String) As String
    Var ePos As Integer

    ePos = InStr(StartStr, Chr$(0))

    If ePos Then
        TrimNull = Left$(StartStr, ePos - 1)
        Exit Function
    End If

    TrimNull = StartStr
End Function

'=====================================
'= フルパス名からファイル名を取り出す
'=====================================
Declare Function GetFileNameFromPath(sFullPath As String) As String
Function GetFileNameFromPath(sFullPath As String) As String
    Var wfd As WIN32_FIND_DATA
    Var hFile As Long
    Var Ret As Long

    hFile = Api_FindFirstFile(sFullPath, wfd)

    If hFile <> INVALID_HANDLE_VALUE Then
        GetFileNameFromPath = TrimNull(wfd.cFileName)
        Ret = Api_FindClose(hFile)
    End If
End Function

'=====================================
'= シェルドロップされたファイル名を取得
'=====================================
```

```
Declare Sub Edit1_DropFiles edecl (ByVal DF As Long)
Sub Edit1_DropFiles(ByVal DF As Long)
    Var CN As Long

    CN = GetDropFileCount(DF)
    FullPath = GetDropFileName(DF, 0)
    Edit1.SetWindowText FullPath
End Sub

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    FullPath = Edit1.GetWindowText
    Text1.SetWindowText GetFileNameFromPath(FullPath)
End Sub

'=======================================
'=
'=======================================
While 1
    WaitEvent
Wend
Stop
End
```

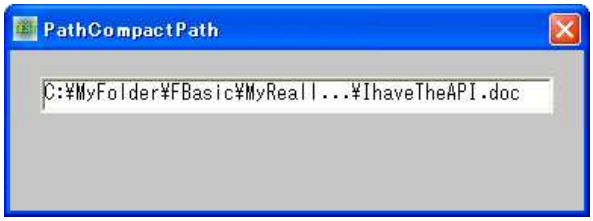
パス名の操作(Ⅰ)

Pathの表示領域の横幅が CtlWidth のとき、この表示領域に収まるように縮めた形式のパスに変換します。
PathCompactPath 短縮形式のパスを取得

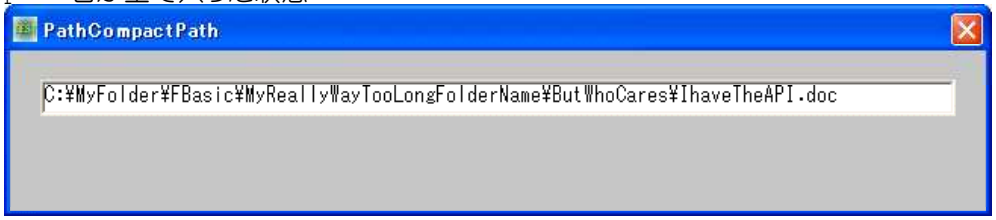
例では、フォームサイズとテキストボックスのサイズを一定の比率で合わせ、フォームサイズを拡大縮小させた場合のパス名の表示を確認しています。
TextBoxとTimerを貼り付けます。



初期状態(短縮部分は...で表されています)。徐々にフォーム幅を拡大



path名が全て入った状態



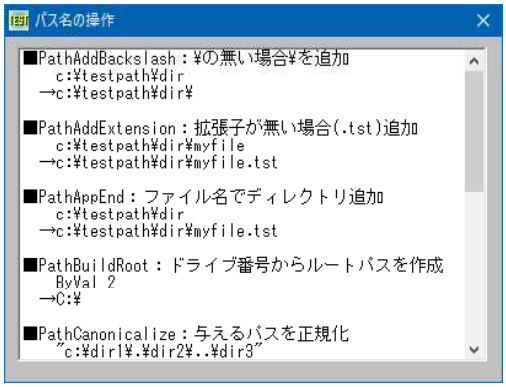

```
'=======================================
'= 短縮形式のパスを取得
'=    (PathCompactPath.bas)
'=======================================
#include "Windows.bi"
```

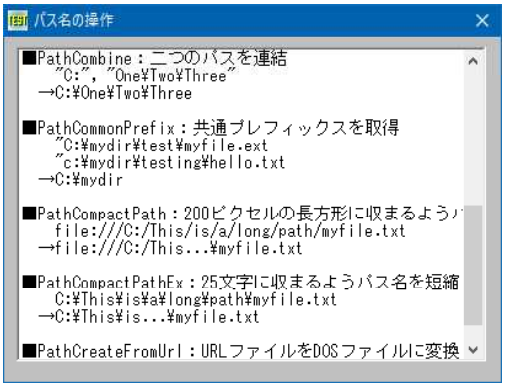
```
' 短縮形式のパスを取得
Declare Function Api_PathCompactPath& Lib "shlwapi" Alias "PathCompactPathA" (ByVal
hDC&, ByVal lpszPath$, ByVal dx&)

' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

Var Shared MainForm As Object
Var Shared Text1 As Object
Var Shared Timer1 As Object

MainForm.Attach GethWnd
Text1.Attach GetDlgItem("Text1")
Timer1.Attach GetDlgItem("Timer1")

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()

    Timer1.SetInterval 100
    Timer1.Enable -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var hDC As Long
    Var CtlWidth As Long
    Var FileName As String

    FileName =
"C:¥MyFolder¥FBasic¥MyReallyWayTooLongFolderName¥ButWhoCares¥IhaveTheAPI.doc"

    ' Textの幅 = MainForm幅 - (左右マージン + Frame幅)
    Text1.SetWindowSize GetWidth - (20 * 2 + 8), 24
    CtlWidth = Text1.GetWidth
    hDC = Api_GetDC(Text1.GethWnd)

    Ret = Api_PathCompactPath(hDC, FileName, CtlWidth )

    Text1.SetWindowText FileName

    Ret = Api_ReleaseDC(hDC, Text1.GethWnd)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

パス名の各種操作(II)

パス名操作に関するAPIをテストしています。
PathAddBackslash パスを表す文字列の最後にバックスラッシュ¥を付け加える。含まれる場合は付け加えられない
PathAddExtension パスを表す文字列の最後にファイル拡張子を付け加える。既に付けられている場合は付加されない

PathAppend 既存のパスにファイル名を追加
PathBuildRoot ドライブ番号からルートパスを生成
PathCanonicalize "."や".."などの文字を含むパス名を、これらの文字を含まないパス名に変換
PathCombine ディレクトリ名とファイルパスの結合
PathCommonPrefix 二つのパスで共通しているプレフィックスを取得
PathCompactPath 短縮形式のパスを取得
PathCompactPathEx 文字数に収まるようにパスの一部を省略
PathCreateFromUrl URLパスを取得しDOSパスに変換






```
'=====================================
'= パス名の操作(II)
'=    (PathFunctions2.bas)
'=====================================
#include "Windows.bi"

' パスを表す文字列の最後にバックスラッシュ¥を付け加える。含まれる場合は付け加えられない
Declare Function Api_PathAddBackslash& Lib "shlwapi" Alias "PathAddBackslashA" (ByVal
pszPath$)

' パスを表す文字列の最後にファイル拡張子を付け加える。既に付けられている場合は付加されない
Declare Function Api_PathAddExtension& Lib "shlwapi" Alias "PathAddExtensionA" (ByVal
pszPath$, ByVal pszExt$)

' 既存のパスにファイル名を追加
Declare Function Api_PathAppend& Lib "shlwapi" Alias "PathAppendA" (ByVal pszPath$,
ByVal pMore$)

'ドライブ番号からルートパスを生成
Declare Function Api_PathBuildRoot& Lib "shlwapi" Alias "PathBuildRootA" (ByVal szRoot$,
ByVal iDrive$)

' "."や".."などの文字を含むパス名を、これらの文字を含まないパス名に変換
Declare Function Api_PathCanonicalize& Lib "shlwapi" Alias "PathCanonicalizeA" (ByVal
pszBuf$, ByVal pszPath$)

' ディレクトリ名とファイルパスの結合
Declare Function Api_PathCombine& Lib "shlwapi" Alias "PathCombineA" (ByVal szDest$,
ByVal lpszDir$, ByVal lpszFile$)

' 二つのパスで共通しているプレフィックスを取得
Declare Function Api_PathCommonPrefix& Lib "shlwapi" Alias "PathCommonPrefixA" (ByVal
pszFile1$, ByVal pszFile2$, ByVal achPath$)

' 短縮形式のパスを取得
Declare Function Api_PathCompactPath& Lib "shlwapi" Alias "PathCompactPathA" (ByVal
hDC&, ByVal pszPath$, ByVal dx&)

' 文字数に収まるようにパスの一部を省略
Declare Function Api_PathCompactPathEx& Lib "shlwapi" Alias "PathCompactPathExA" (ByVal
pszOut$, ByVal pszSrc$, ByVal cchMax&, ByVal dwFlags&)

' URLパスを取得しDOSパスに変換
Declare Sub Api_PathCreateFromUrl Lib "shlwapi" Alias "PathCreateFromUrlA" (ByVal
pszUrl$, ByVal pszPath$, ByRef pcchPath&, ByVal dwFlags&)
```

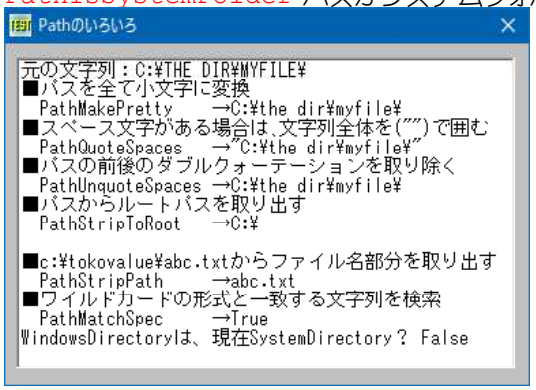
```
' 指定されたウィンドウのクライアント領域または画面全体を表すディスプレイデバイスコンテキストのハンドルを取得
Declare Function Api_GetDC& Lib "user32" Alias "GetDC" (ByVal hWnd&)

' デバイスコンテキストを解放
Declare Function Api_ReleaseDC& Lib "user32" Alias "ReleaseDC" (ByVal hWnd&, ByVal hDC&)

Var Shared List1 As Object
List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
List1.SetWindowSize 382, 256

'=====================================
'=
'=====================================
Declare Function StripTerminator(sInput As String) As String
Function StripTerminator(sInput As String) As String
    Var ZeroPos As Long

    ZeroPos = instr(1, sInput, Chr$(0))
    If ZeroPos > 0 Then
        StripTerminator = Left$(sInput, ZeroPos - 1)
    Else
        StripTerminator = sInput
    End If
End Function

'=====================================
'= パス名の各種操作
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var sSave As String
    Var hDC As Long
    Var Ret As Long

    hDC = Api_GetDC(List1.GethWnd)

    List1.ResetContent

    'パスのバッファ
    sSave = "c:¥testpath¥dir" + String$(100, Chr$(0))
    List1.Addstring "■PathAddBackslash:¥の無い場合¥を追加"
    List1.AddString "    " & sSave

    '¥の無い場合¥を追加
    Ret = Api_PathAddBackslash(sSave)
    List1.AddString "  →" & StripTerminator(sSave)              '結果
    List1.AddString ""

    '拡張子のないファイル名とバッファ
    sSave = "c:¥testpath¥dir¥myfile" + String$(100, Chr$(0))
    List1.Addstring "■PathAddExtension:拡張子が無い場合(.tst)追加"
    List1.AddString "    " & sSave

    '拡張子を持っていない場合(.tst)追加
    Ret = Api_PathAddExtension(sSave, ".tst")
    List1.AddString "  →" & StripTerminator(sSave)              '結果
    List1.AddString ""

   'ディレクトリバッファ
    sSave = "c:¥testpath¥dir" + String$(100, Chr$(0))
    List1.Addstring "■PathAppend:ファイル名でディレクトリ追加"
    List1.AddString "    " & sSave

    'ファイル名でディレクトリ追加
    Ret = Api_PathAppend(sSave, "myfile.tst")
    List1.AddString "  →" & StripTerminator(sSave)              '結果
    List1.AddString ""

    'バッファ
    sSave = String$(100, Chr$(0))
```

```
    List1.Addstring "■PathBuildRoot:ドライブ番号からルートパスを作成"
    List1.AddString "    ByVal 2"

    '与えられたドライブ番号からルートパスを作成
    Ret = Api_PathBuildRoot(sSave, ByVal 2)
    List1.AddString "  →" & StripTerminator(sSave)              '結果
    List1.AddString ""

    'バッファ
    sSave = String$(255, Chr$(0))
    List1.Addstring "■PathCanonicalize:与えるパスを正規化"
    List1.AddString "    " & Chr$(34) & "c:¥dir1¥.¥dir2¥..¥dir3" & Chr$(34)

    '与えるパスを正規化(Canonicalize)
    Ret = Api_PathCanonicalize(sSave, "c:¥dir1¥.¥dir2¥..¥dir3")
    List1.AddString "  →" & StripTerminator(sSave)              '結果
    List1.AddString ""

    'バッファ
    sSave = String$(255, Chr$(0))
    List1.Addstring "■PathCombine:二つのパスを連結"
    List1.AddString "    " & Chr$(34) & "C:" & Chr$(34) & ", " & Chr$(34) & "One¥Two¥Three"
& Chr$(34)

    '二つのパスを連結
    Ret = Api_PathCombine(sSave, "C:", "One¥Two¥Three")
    List1.AddString "  →" & StripTerminator(sSave)              '結果
    List1.AddString ""

    'バッファ･/ 二つのパスで共通しているプレフィックスを取得
    sSave = String$(255, Chr$(0))
    List1.Addstring "■PathCommonPrefix:共通プレフィックスを取得"
    List1.AddString "    " & Chr$(34) & "C:¥mydir¥test¥myfile.ext"
    List1.AddString "    " & Chr$(34) & "c:¥mydir¥testing¥hello.txt"
    Ret = Api_PathCommonPrefix("C:¥mydir¥test¥myfile.ext",
"c:¥mydir¥testing¥hello.txt", sSave)
    List1.AddString "  →" & StripTerminator(sSave)              '結果
    List1.AddString ""
    sSave = "file:///C:/This/is/a/long/path/myfile.txt"
    List1.Addstring "■PathCompactPath:200ピクセルの長方形に収まるようパスを短縮"
    List1.AddString "    " & sSave

    '例では200ピクセルの長方形に収まるようパスを短縮
    Ret = Api_PathCompactPath(hDC, sSave, 200)
    List1.AddString "  →" & StripTerminator(sSave)              '結果
    List1.AddString ""

    'バッファ
    sSave = String$(255, Chr$(0))
    List1.Addstring "■PathCompactPathEx:25文字に収まるようパス名を短縮"
    List1.AddString "    C:¥This¥is¥a¥long¥path¥myfile.txt"

     '25文字に収まるようパス名を短縮
    Ret = Api_PathCompactPathEx(sSave, "C:¥This¥is¥a¥long¥path¥myfile.txt", 25, 0)
    List1.AddString "  →" & StripTerminator(sSave)              '結果
    List1.AddString ""

    'バッファ
    sSave = String$(255, Chr$(0))
    List1.Addstring "■PathCreateFromUrl:URLファイルをDOSファイルに変換"
    List1.AddString "    file:///C:/dir/myfile.ext"

    'URLファイルをDOSファイルに変換
    Api_PathCreateFromUrl "file:///C:/dir/myfile.ext", sSave, Len(sSave), 0
    List1.AddString "  →" & StripTerminator(sSave)              '結果

    Ret = Api_ReleaseDC(hDC, List1.GethWnd)
End Sub
```

```
'====================================
'=
'====================================
While 1
    WaitEvent
Wend
Stop
End
```

パス名の操作(III)

パスに関する機能をテストします。
PathMakePretty 大文字のパスを小文字に変換
PathQuoteSpaces パスを表す文字列にスペース文字がある場合は、文字列全体をダブルクォーテーションマークで囲む
PathUnquoteSpaces パスの前後のダブルクォーテーションを取り除く
PathStripPath パスからファイル名部分を取り出す
PathStripToRoot パスからルートパスを取り出す
PathMatchSpec ワイルドカードの形式と一致する文字列を検索
PathMakeSystemFolder システムフォルダのファイル属性を設定
PathIsSystemFolder パスがシステムフォルダであるかどうか判定




```
'====================================
'= パス名の操作(III)
'=    (PathFunction.bas)
'====================================
#include "Windows.bi"

' 大文字のパスを小文字に変換
Declare Function Api_PathMakePretty& Lib "shlwapi" Alias "PathMakePrettyA" (ByVal
pszPath$)

' パスを表す文字列にスペース文字がある場合は、文字列全体をダブルクォーテーションマークで囲む
Declare Sub Api_PathQuoteSpaces Lib "shlwapi" Alias "PathQuoteSpacesA" (ByVal lpsz$)

' パスの前後のダブルクォーテーションを取り除く
Declare Sub Api_PathUnquoteSpaces Lib "shlwapi" Alias "PathUnquoteSpacesA" (ByVal lpsz$)

' パスからファイル名部分を取り出す
Declare Sub Api_PathStripPath Lib "shlwapi" Alias "PathStripPathA" (ByVal pszPath$)

' パスからルートパスを取り出す
Declare Function Api_PathStripToRoot& Lib "shlwapi" Alias "PathStripToRootA" (ByVal
pszPath$)

' ワイルドカードの形式と一致する文字列を検索
Declare Function Api_PathMatchSpec& Lib "shlwapi" Alias "PathMatchSpecA" (ByVal
pszFile$, ByVal pszSpec$)

' システムフォルダのファイル属性を設定
Declare Function Api_PathMakeSystemFolder& Lib "shlwapi" Alias "PathMakeSystemFolderA"
(ByVal pszPath$)
```

```
' パスがシステムフォルダであるかどうか判定
Declare Function Api_PathIsSystemFolder& Lib "shlwapi" Alias "PathIsSystemFolderA"
(ByVal pszPath$, ByVal dwAttrb&)

Var Shared List1 As Object
List1.Attach GetDlgItem("List1") : List1.SetFontSize 14

'=====================================
'=    Chr$(0)の除去
'=====================================
Declare Function StripTerminator(sInput As String) As String
Function StripTerminator(sInput As String) As String
    Var ZeroPos As Long
    ZeroPos = InStr(1, sInput, Chr$(0))
    If ZeroPos > 0 Then
        StripTerminator = Left$(sInput, ZeroPos - 1)
    Else
        StripTerminator = sInput
    End If
End Function

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var sSave As String
    Var State As String
    Var Ret As Long

    sSave = "C:¥THE DIR¥MYFILE¥" & String$(100, 0)
    List1.AddString "元の文字列:" & StripTerminator(sSave)

    Ret = Api_PathMakePretty(sSave)
    List1.AddString "■パスを全て小文字に変換"
    List1.AddString "  PathMakePretty" & "    →" & StripTerminator(sSave)

    Api_PathQuoteSpaces sSave
    List1.AddString "■スペース文字がある場合は、文字列全体を(" & Chr$(&H22) & Chr$(&H22) & ")で
囲む"
    List1.AddString "  PathQuoteSpaces" & "   →" & StripTerminator(sSave)

    Api_PathUnquoteSpaces sSave
    List1.AddString "■パスの前後のダブルクォーテーションを取り除く"
    List1.AddString "  PathUnquoteSpaces" & " →" & StripTerminator(sSave)

    Ret = Api_PathStripToRoot(sSave)
    List1.AddString "■パスからルートパスを取り出す"
    List1.AddString "  PathStripToRoot" & "   →" & StripTerminator(sSave)
    List1.AddString ""

    sSave = "c:¥tokovalue¥abc.txt"
    List1.AddString "■" & sSave & "からファイル名部分を取り出す"
    Api_PathStripPath sSave
    List1.AddString "  PathStripPath" & "     →"& StripTerminator(sSave)

    List1.AddString "■ワイルドカードの形式と一致する文字列を検索"
    Ret = Api_PathMatchSpec("c:¥dir¥file.ext", "*.e?t")
    If Ret = 0 Then State = "False" else State = "True"
    List1.AddString "  PathMatchSpec" & "     →" & State

    'SystemFolderになるように適切な属性をWindowsDirectoryに与える
    Ret = Api_PathMakeSystemFolder("c:¥windows")
    Ret = Api_PathIsSystemFolder("c:¥windows", 0)
    If Ret = 0 Then State = "False" else State = "True"

    '成功したかどうかのチェック
    List1.AddString "WindowsDirectoryは、現在SystemDirectory？ " & State
End Sub
```

```
'====================================
'=
'====================================
While 1
    WaitEvent
Wend
Stop
End
```