"."や".."などの文字を含まないパス名に変換

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


```
'=======================================
'= "."や".."などの文字を含まないパス名に変換
'=    (PathCanonicalize.bas)
'=======================================
#include "Windows.bi"

' "."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$, ByVal pszPath$)

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:¥dir1¥.¥dir2¥..¥dir3"
End Sub

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

    Buff = String$(100, 0)
    sSave = Edit1.GetWindowText

    Ret = Api_PathCanonicalize(Buff, sSave)

    Text1.SetWindowText TrimNull(Buff)
```

```
End Sub

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

64Bit版のx86エミュレータで動作しているかの判別(ＩＩ)

IsWow64Process 指定されたプロセスの実行に、WOW64 が使用されているかどうかを決定
GetCurrentProcess 現在のプロセスに対応する疑似ハンドルを取得
GetNativeSystemInfo WOW64 で実行中のアプリケーションシステムに関する情報を取得


```
'=====================================
'= 64Bit版のx86エミュレータで動作しているかの判別(ＩＩ)
'=    (IsWow64Process.bas)
'=====================================
#include "Windows.bi"

Type SYSTEM_INFO
    dwOemId                     As Long         'このメンバは使われない
    dwPageSize                  As Long         'メモリページのサイズ
    lpMinimumApplicationAddress As Long         'アプリケーションが利用可能なメモリ空間の最下位アドレス
    lpMaximumApplicationAddress As Long         'アプリケーションが利用可能なメモリ空間の最上位アドレス
    dwActiveProcessorMask       As Long         'システム中に存在するプロセッサのビットマスク
    dwNumberOfProcessors        As Long         'システム中に存在するプロセッサの数
    dwProcessorType             As Long         'プロセッサの種類(386･486･586)
    dwAllocationGranularity     As Long         'メモリ空間割り当ての最小単位
    wProcessorLevel             As Integer      'プロセッサの種類(WindowsNT系OSのみ)
    wProcessorRevision          As Integer      'プロセッサのバージョン(WindowsNT系OSのみ)
End Type

' 指定されたプロセスの実行に、WOW64 が使用されているかどうかを決定
Declare Function Api_IsWow64Process& Lib "kernel32" Alias "IsWow64Process" (ByVal
hProcess&, ByRef Wow64Process&)

' 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" ()

'WOW64 で実行中のアプリケーションシステムに関する情報を取得
Declare Sub Api_GetNativeSystemInfo Lib "kernel32" Alias "GetNativeSystemInfo"
(lpSystemInfo As SYSTEM_INFO)

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

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl()
Sub Button1_on()
    Var Value As Long
    Var Ret As Long
    Ret = Api_IsWow64Process(GetCurrentProcess, Value)
```

```
    If Value = 0 Then
        A% = MessageBox("", "このアプリケーションは、" & Chr$(13, 10) & "64Bitコンピューターの" &
Chr$(13, 10) & "x86エミュレーションで" & Chr$(13, 10) & "動作していません！", 0, 2)

    Else
        Var si64 As SYSTEM_INFO
        Api_GetNativeSystemInfo si64
        A% = MessageBox("", "64ビットシステム上の" & Chr$(13, 10) & "プロセッサの数: " & Chr$(13,
10) & Str$(si64.dwNumberOfProcessors), 0, 2)
    End If
End Sub

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

64Bit版のx86エミュレータで動作しているかの判断(II)

IsWow64Process 指定されたプロセスの実行に、WOW64 が使用されているかどうかを決定
GetModuleHandle 指定の実行モジュールのハンドルを取得
GetProcAddress 実行モジュール内の関数アドレスを取得
GetCurrentProcess 現在のプロセスに対応する疑似ハンドルを取得

Windows 7 Professional SP1(32Bit版)の例　　Windows 7 Home Premium SP1(64Bit版)の例


```
'=====================================
'= 64Bit版のx86エミュレータで動作しているかの判別(II)
'=    (IsWow64Process2.bas)
'=====================================
#include "Windows.bi"

' 指定されたプロセスの実行に、WOW64 が使用されているかどうかを決定
Declare Function Api_IsWow64Process& Lib "kernel32" Alias "IsWow64Process" (ByVal
hProcess&, ByRef Wow64Process&)

' 指定の実行モジュールのハンドルを取得
Declare Function Api_GetModuleHandle& Lib "Kernel32" Alias "GetModuleHandleA" (ByVal
ModuleName$)

' 実行モジュール内の関数アドレスを取得
Declare Function Api_GetProcAddress& Lib "Kernel32" Alias "GetProcAddress" (ByVal
ModuleHandle&, ByVal ProcName$)

' 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" ()

Var Shared Button1 As Object

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

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

```
Declare Function IsWow64() As Integer
Function IsWow64() As Integer
    Var Wow64 As Long
    Var Wow64Proc As Long

    Wow64 = False
    Wow64Proc = Api_GetProcAddress(Api_GetModuleHandle("kernel32"), "Wow64Process")
    If (0 <> Wow64Proc) Then
        If 0 = Api_Wow64Process(Api_GetCurrentProcess(), Wow64) Then
           ' handle error
        End If
    End If
    IsWow64 = (Wow64 <> 0)
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl()
Sub Button1_on()
    If IsWow64() = True then
        A% = MessageBox("", "Windows x64 で動作しています！", 0, 2)
    Else
        A% = MessageBox("", "Windows x64 で動作していません！", 0, 2)
    End If
End Sub

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

AddStringの速度比較

コンボボックス、リストボックスにAddStringする速度を比較テストします。
LockWindowUpdate 再描画しようとするウィンドウをロックする
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
SendMessage ウィンドウにメッセージを送信

F-BasicでのAddStringと、SendNessageでCB_ADDSTRING(LB_ADDSTRING)を送った場合の速度ををを比較してみます。
例では、5000までの数値を(昇順・降順は関係ありません)AddStringしています。
それぞれ、リスト表示しながら読み込んだ場合、リスト非表示で読み込み込んだ場合、読み込み完了まで再描画を停止した場合の時間を計測します。
ComboBox
F-Basicの場合

SendNessageでCB ADDSTRINGを送った場合

ListBox(リストボックスを非表示した状態‥画面体裁上List1にAddStringしている間、List2「ダミー」を表示させています。)

F-Basicの場合

SendNessageでLB ADDSTRINGを送った場合

※timeGetTime
この関数と timeGetSystemTime関数との唯一の違いは、timeGetSystemTime関数では、システム時刻を返すのにMMTIME構造体を使う点です。timeGetTime関数では、timeGetSystemTime関数よりもオーバーヘッドが少なくて済みます。
timeGetTime関数で返される値は DWORD 値であることに注意してください。戻り値は0 ミリ秒から 2^32 ミリ秒の間を循環します。2^32 ミリ秒は約 49.71 日にあたります。
計算にtimeGetTime関数の戻り値を直接使うようなコードでは、特にコードの実行を制御する場合などに、問題が発生する可能性があります。計算では常に、2つの timeGetTime関数の戻り値の差を使います。
Windows NT:timeGetTime関数の既定の精度は、マシンによっては5 ミリ秒以上になる場合があります。
timeGetTime関数の精度は、timeBeginPeriod関数とtimeEndPeriod関数を使って上げることができます。こうすると、2 つの連続したtimeGetTime関数の戻り値の差の最小値は、timeBeginPeriod関数とtimeEndPeriod関数を使って設定された間隔の最小値と同じになります。高い分解能で短い時間間隔を測定するには、QueryPerformanceCounter関数とQueryPerformanceFrequency関数を使います。
Windows 95:timeGetTime関数の既定の精度は1 ミリ秒です。
つまり、timeGetTime関数は、1 ミリ秒しか違わない連続値を返すことができます。timeBeginPeriod関数とtimeEndPeriod関数に対して行われた呼び出しがどのような性質のものであっても、これは変わりません。
<SDKより抜粋>

```
'=======================================
'= AddString速度比較
'=======================================
#include "Windows.bi"

' 再描画しようとするウィンドウをロックする
Declare Function Api_LockWindowUpdate& Lib "user32" Alias "LockWindowUpdate" (ByVal
hwndLock&)

' 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" ()
```

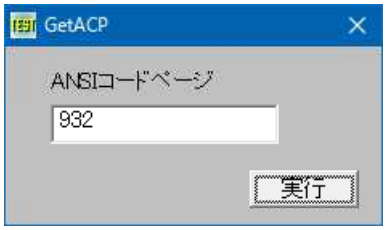
```
' アプリケーションまたはデバイスドライバの最小タイマ分解能を設定
Declare Function Api_timeBeginPeriod& Lib "winmm" Alias "timeBeginPeriod" (ByVal
uPeriod&)

' 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&)

' 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_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_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 LB_ADDSTRING &H180                 'リストボックスに文字列を追加する

Var Shared Combo1 As Object
Var Shared Text1 As Object
Var Shared List(1) As Object
Var Shared Check(1) As Object
Var Shared Radio(1) As Object

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

For i = 0 To 1
    List(i).Attach GetDlgItem("List" & Trim$(Str$(i + 1)))
    Check(i).Attach GetDlgItem("Check" & Trim$(Str$(i + 1)))
    Radio(i).Attach GetDlgItem("Radio" & Trim$(Str$(i + 1)))

    List(i).SetFontSize 14
    Check(i).SetFontSize 14
    Radio(i).SetFontSize 14
Next

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

'=====================================
'=  F-BASIC による項目書き込み
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Item As String
    Var i As Integer
    Var nStart As Long
    Var nEnd As Long
    Var Ret As Long

    Combo1.ResetContent
    List(0).ResetContent
    Text1.SetWindowText ""

    If Check(0).GetCheck = 1 Then

        'ComboBox･ListBox非表示
        List(1).ShowWindow -1 : List(0).ShowWindow 0
    Else If Index = 0 Then
        Ret = Api_SendMessage(Combo1.GethWnd, CB_SHOWDROPDOWN, 1, ByVal 0)
    End If

    Ret = Api_timeBeginPeriod(1)
    nStart = Api_timeGetTime()
```

```
    'コンボボックス
    If Index = 0 Then
        If Check(1).GetCheck = 1 Then Ret = Api_LockWindowUpdate(Combo1.GethWnd)

        For i = 5000 To 1 step -1
            Item = Str$(i)
            Combo1.AddString Item
        Next

    'リストボックス
    Else

        '再描画を停止
        If Check(1).GetCheck = 1 Then Ret = Api_LockWindowUpdate(List(0).GethWnd)
        For i = 5000 To 1 step -1
            Item = Str$(i)
            List(0).AddString Item
        Next
    End If

    '再描画を解放
    Ret = Api_LockWindowUpdate(0)

    nEnd = Api_timeGetTime()
    Ret = Api_timeEndPeriod(1)

    List(1).ShowWindow 0
    List(0).ShowWindow -1

    Text1.SetWindowtext Str$((Cdbl(nEnd) - Cdbl(nStart)) / 1000) & " 秒"
End Sub

'=====================================
'= Win32 API による項目書き込み
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Item As String
    Var i As Integer
    Var nStart As Long
    Var nEnd As Long
    Var Ret As Long

    Combo1.ResetContent
    List(0).ResetContent
    Text1.SetWindowText ""

    If Check(0).GetCheck = 1 Then

        'ComboBox･ListBox非表示
        List(1).ShowWindow -1 : List(0).ShowWindow 0
    Else If Index = 0 Then

        'ComboBoxプルダウン表示
        Ret = Api_SendMessage(Combo1.GethWnd, CB_SHOWDROPDOWN, 1, ByVal 0)
    End If

    Ret = Api_timeBeginPeriod(1)
    nStart = Api_timeGetTime()

    'コンボボックス
    If Index = 0 Then
        If Check(1).GetCheck = 1 Then Ret = Api_LockWindowUpdate(Combo1.GethWnd)
        For i = 5000 To 1 step -1
            Item = Str$(i)
            Ret = Api_SendMessage(Combo1.GethWnd, CB_ADDSTRING, 0, Item)
        Next
        Ret = Api_SendMessage(Combo1.GethWnd, CB_SHOWDROPDOWN, 1, ByVal 0)
```

```
    'リストボックス
    Else

        '再描画を停止
        If Check(1).GetCheck = 1 Then Ret = Api_LockWindowUpdate(List(0).GethWnd)
        For i = 5000 To 1 step -1
            Item = Str$(i)
            Ret = Api_SendMessage(List(0).GethWnd, LB_ADDSTRING, 0, Item)
        Next
    End If

    '再描画解放
    Ret = Api_LockWindowUpdate(0)

    nEnd = Api_timeGetTime()
    Ret = Api_timeEndPeriod(1)

    List(1).ShowWindow 0
    List(0).ShowWindow -1

    Text1.SetWindowtext Str$((Cdbl(nEnd) - Cdbl(nStart)) / 1000) & " 秒"
End Sub

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

ANSIコードページの識別子を取得

GetACP システムで現在有効になっている ANSI コードページ(**A**NSI **C**ode**P**age)の識別子を取得



参考
ANSI コードページの識別子は以下の通りです。

識別子	意味
874	タイ語
932	日本語
936	中国語簡体字(中国、シンガポール)
949	韓国語
950	中国語繁体字(台湾、香港)
1200	Unicode(ISO 10646 の BMP)
1250	Windows 3.1 東欧
1251	Windows 3.1 キリル文字
1252	Windows 3.1 ラテン I(米国、西欧)
1253	Windows 3.1 ギリシャ語
1254	Windows 3.1 トルコ語
1255	ヘブライ語
1256	アラビア語
1257	バルト語族

```
'=======================================
'= ANSIコードページの識別子を取得
'=    (GetACP.bas)
'=======================================
#include "Windows.bi"
```

```
' システムで現在有効になっている ANSI コードページの識別子を取得
Declare Function Api_GetACP& Lib "kernel32" Alias "GetACP" ()

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

    Ret = Api_GetACP()

    Text2.SetWindowText Str$(Ret)
End Sub

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

ANSI文字列をUnicode文字列に変換

ANSI(1バイトASCII文字、2バイトマルチバイト文字:漢字)文字列を、全て2バイトで表現するUnicode文字列に変換します。
DoFileDownload ファイルのダウンロードを呼び出す
MultiByteToWideChar ANSI文字列をUnicode文字列に変換

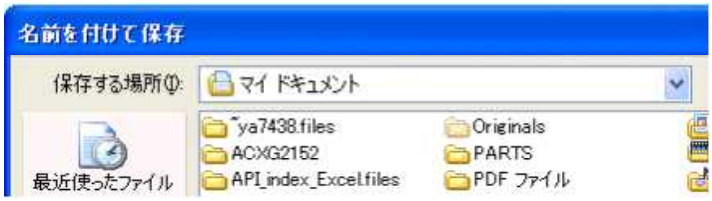
VisualBasicでのStrConv(ANSI文字列, vbUnicode)をAPIで実行します。保存された文字列をTeraPadで読み込み表示させています。
全て半角英数の場合

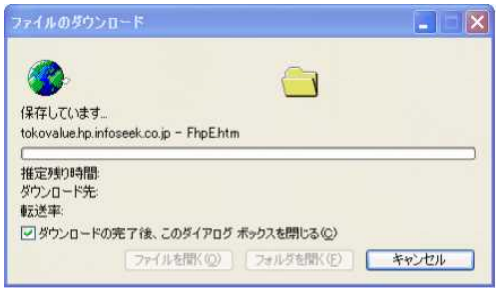
ダンプで見てみる

```
00 : 68 74 74 70 3A 2F 2F 74  6F 6B 6F 76 61 6C 75 65 : http://tokovalue
10 : 2E 68 70 2E 69 6E 66 6F  73 65 65 6B 2E 63 6F 2E : .hp.infoseek.co.
20 : 6A 70 2F 49 6E 64 65 78  2E 68 74 6D 68 00 74 00 : jp/Index.htmh.t.
30 : 74 00 70 00 3A 00 2F 00  2F 00 74 00 6F 00 6B 00 : t.p.:././.t.o.k.
40 : 6F 00 76 00 61 00 6C 00  75 00 65 00 2E 00 68 00 : o.v.a.l.u.e...h.
50 : 70 00 2E 00 69 00 6E 00  66 00 6F 00 73 00 65 00 : p...i.n.f.o.s.e.
60 : 65 00 6B 00 2E 00 63 00  6F 00 2E 00 6A 00 70 00 : e.k...c.o...j.p.
70 : 2F 00 49 00 6E 00 64 00  65 00 78 00 2E 00 68 00 : /.I.n.d.e.x...h.
80 : 74 00 6D 00                                     : t.m.
```

漢字が含まれている場合

DoFileDownloadを実行すると、VisualBasicでは図のダイアログが表示されますが、F-Basicでは対応していないのか、下記の警告メッセージが表示されます。





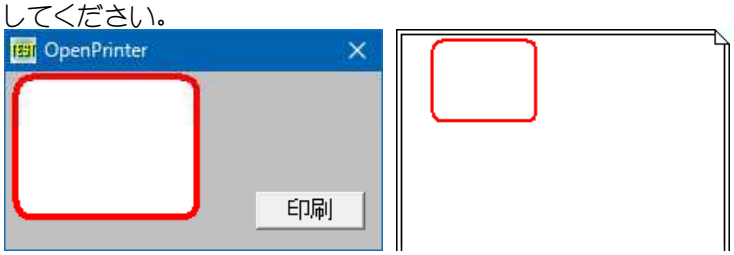

```
'=======================================
'= ANSI文字列をUniCode文字列に変換
'=    (MultiByteToWideChar.bas)
'=======================================
#include "Windows.bi"

' ファイルのダウンロード
Declare Function Api_DoFileDownload& Lib "shdocvw" Alias "DoFileDownload" (ByVal
lpszFile$)

' 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&)

    '* CodePage       ANSI code page
    '* dwFlags        既定の変換方法
    '* lpMultiByteStr 変換元文字列
    '* cchMultiByte   変換元文字列サイズ(バイト数で指定:-1指定で自動計算)
    '* lpWideCharStr  変換先バッファ
    '* cchWideChar    変換先バッファサイズ(文字数で指定:0指定で必要なバッファサイズを返す)

#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
Var Shared Edit1 As Object
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14

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

    Buff = String$(256, Chr$(0))
    Buff = Edit1.GetWindowText
    wBuff = String$(256, Chr$(0))

    '必要なバッファサイズを取得
    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)

    Ret = Api_DoFileDownload(wBuff)

    'バイナリダンプで確認の為ファイルに保存
    Open "test.txt" For BinIO As #1
        FWrite #1, Buff
        FWrite #1, wBuff
    Close
End Sub

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

APIを使った印刷(RoundRect)

APIを使って印刷をします。
OpenPrinter プリンタオブジェクトをオープン
CreateDC 指定されたデバイスコンテキストを、指定された名前で作成
CreatePenIndirect LOGFONTを定義して論理ペンを作成
RoundRect 角の丸い矩形を描画
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
DeleteObject 論理オブジェクトを削除し、関連づけされた全てのリソースを解放
StartDoc 印刷ジョブの開始
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ページ書き込み終了を通知
EndDoc 印刷ジョブの終了
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の解放

左:フォーム上に描画した矩形を...　　右:各数値を10倍して印刷した状態(適当に縮小しています)
例では、プリンタ名を明示的に指定しています。プリンタ名の取得については「プリンタの対応する用紙を取得」を参照してください。




```
'=====================================
'= APIを使った印刷
'=    (PrintRoundRect.bas)
'=====================================
#include "Windows.bi"

Type DOCINFO
    cbSize              As Long
    lpszDocName         As Long
    lpszOutput          As Long
End Type

Type PRINTER_DEFAULTS
    pDatatype           As Long
    pDevMode            As Long
    DesiredAccess       As Long
End Type
```

```
Type POINTAPI
    x   As Long
    y   As Long
End Type

Type LOGPEN
    lopnStyle  As Long
    lopnWidth  As POINTAPI
    lopnColor  As Long
End Type

' プリンタオブジェクトをオープン
Declare Function Api_OpenPrinter& Lib "winspool.drv" Alias "OpenPrinterA" (ByVal
pPrinterName$, phPrinter&, pDefault As PRINTER_DEFAULTS)

' 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$, ByVal
lpDevName$, ByVal lpOutput$, ByVal lpInitData&)

' LOGPEN構造体を定義して論理ペンを作成
Declare Function Api_CreatePenIndirect& Lib "gdi32" Alias "CreatePenIndirect" (lpLogPen
As LOGPEN)

' 角の丸い矩形を描画
Declare Function Api_RoundRect& Lib "gdi32" Alias "RoundRect" (ByVal hDC&, ByVal
nLeftRect&, ByVal nTopRect&, ByVal nRightRect&, ByVal nBottomRect&, 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&)

' 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&)

#define PS_SOLID 0
#define PRINTER_ACCESS_ADMINISTER &H4      'プリンタアクセス権の管理者権限を示す定数の宣言
#define PRINTER_ACCESS_USE &H8             'プリンタアクセス権のユーザー権限を示す定数の宣言

#define vbWhite &HFFFFFF                   '白のカラーコード
#define vbBlack &H000000                   '黒のカラーコード
#define vbBlue &HFF0000                    '青のカラーコード
#define vbRed &H0000FF                     '赤のカラーコード
#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 lp As LOGPEN
Var Shared pa As POINTAPI
```

```
Var Shared hndPen As Long
Var Shared oldPen As Long
Var Shared x1 As Long
Var Shared y1 As Long
Var Shared x2 As Long
Var Shared y2 As Long
Var Shared x3 As Long
Var Shared y3 As Long
Var Shared w As Long

Var Shared Button1 As Object

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

'=====================================
'= フォームに描画
'=====================================
Declare Sub MainForm_MouseMove edecl ()
Sub MainForm_MouseMove()
    Var hDC As Long
    Var Ret As Long

    hDC = Api_GetDC(GethWnd)

    'ペン太さ・色
    w = 4
    pa.x = w
    lp.lopnColor = vbRed
    lp.lopnStyle = PS_SOLID
    lp.lopnWidth = pa

    'ペンオブジェクト作成
    hndPen = Api_CreatePenIndirect(lp)

    'プリンタデバイスコンテキストにペンオブジェクト選択
    oldPen = Api_SelectObject(hDC, hndPen)

    '左上座標
    x1 = 6
    y1 = 3

    '右下座標
    x2 = 120
    y2 = 90

    'コーナー
    x3 = 20
    y3 = 20

    'プリンタデバイスコンテキストに角を丸めた長方形を描画
    Ret = Api_RoundRect(hDC, x1, y1, x2, y2, x3, y3)

    'ペン削除
    Ret = Api_DeleteObject(hndPen)
    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

'=====================================
'= 印刷
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var DevName As String
    Var di As DOCINFO
    Var pd As PRINTER_DEFAULTS
    Var prhDC As Long
    Var ex As Long
    Var Ret As Long

    ex = 10
```

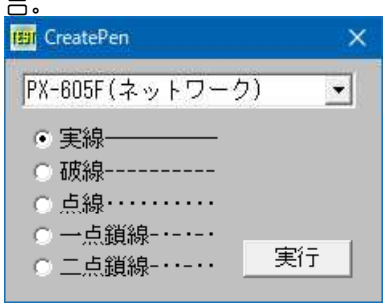
```
    '通常使うプリンタ
    'プリンタ名の取得は、「プリンタの対応する用紙を取得」参照
    DevName = "pdfFactory Pro"

    'プリンタ初期化
    pd.pDatatype = StrAdr(Chr$(0))
    pd.pDevMode = 0
    pd.DesiredAccess = PRINTER_ACCESS_ADMINISTER Or PRINTER_ACCESS_USE

    'プリンタオープン
    Ret = Api_OpenPrinter(DevName, prhDC, pd)

    'プリンタデバイスコンテキストを取得
    prhDC = Api_CreateDC("WINSPOOL", DevName, vbNullString, 0)
    If prhDC = 0 Then GoTo *cleanup

    di.cbSize = Len(di)
    di.lpszOutput = StrAdr(Chr$(0))

    '印刷プロセス開始
    Ret = Api_StartDoc(prhDC, di)
    Ret = Api_StartPage(prhDC)

    'ペン太さ・色
    pa.x = w * ex
    lp.lopnColor = vbRed
    lp.lopnStyle = PS_SOLID
    lp.lopnWidth = pa

    'ペンオブジェクト作成
    hndPen = Api_CreatePenIndirect(lp)

    'プリンタデバイスコンテキストにペンオブジェクト選択
    oldPen = Api_SelectObject(prhDC, hndPen)

    'プリンタデバイスコンテキストに角を丸めた長方形を描画
    Ret = Api_RoundRect(prhDC, x1 * ex, y1 * ex, x2 * ex, y2 * ex, x3 * ex, y3 * ex)

    'ペン削除
    Ret = Api_DeleteObject(hndPen)

    '印刷プロセス終了
    Ret = Api_EndPage(prhDC)
    If Ret >= 0 Then Ret = Api_EndDoc(prhDC)

*cleanup
    If prhDC <> 0 Then Ret = Api_DeleteDC(prhDC)
End Sub

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

APIを使った印刷（文字列・楕円）

EnumPrinters 使用可能なプリンタ・プリントサーバーなどを列挙
CopyMemory ある位置から別の位置にメモリブロックを移動
SetDefaultPrinter 通常使うプリンタの設定
CreateDC 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
GetDeviceCaps デバイス固有の情報を取得

DeleteObject 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
TextOut 文字を描画
Ellipse 楕円の描画
GetTextExtentPoint32 文字列全体の幅と高さを取得
StartDoc 印刷ジョブを開始
StartPage プリンタドライバがデータを受け取る準備
EndDoc 印刷ジョブを終了
EndPage 1ページ書き込みの終了を通知
CreatePen 論理ペンを作成

接続されているプリンタを列挙し、指定したプリンタに指定文字列、楕円を印刷しています。印刷例は実線を指定した場合。




```
'=======================================
'= 文字列･楕円の印刷
'=    (CreatePen.bas)
'=======================================
#include "Windows.bi"

Type POINTAPI
    X   As Long
    Y   As Long
End Type

Type DOCINFO
    cbSize              As Long
    lpszdiName          As Long
    lpszOutput          As Long
    lpszDataType        As Long
    fwType              As Long
End Type

Type PRINTER_INFO_5
    pPrinterName            As Long
    pPortName               As Long
    Attributes              As Long
    DeviceNotSelectedTimeOut As Long
    TransmissionRetryTimeOut As Long
End Type

' 使用可能なプリンタ･プリントサーバーなどを列挙する
Declare Function Api_EnumPrinters& Lib "winspool.drv" Alias "EnumPrintersA" (ByVal
Flags&, ByVal Name$, ByVal Level&, pPrinterEnum As Any, ByVal cbBuf&, pcbNeededed&,
pcReturned&)

' 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&)

' 通常使うプリンタの設定
Declare Function Api_SetDefaultPrinter& Lib "winspool.drv" Alias "SetDefaultPrinterA"
(ByVal pszPrinter$)

' 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&)
```

```
' デバイス固有の情報を取得
Declare Function Api_GetDeviceCaps& Lib "gdi32" Alias "GetDeviceCaps" (ByVal hDC&, 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&)

' 文字を描画
Declare Function Api_TextOut& Lib "gdi32" Alias "TextOutA" (ByVal hDC&, ByVal nXStart&,
ByVal nYStart&, ByVal lpString$, ByVal cbString&)

' 楕円の描画
Declare Function Api_Ellipse& Lib "gdi32" Alias "Ellipse" (ByVal hDC&, ByVal X1&, ByVal
Y1&, ByVal X2&, ByVal Y2&)

' 文字列全体の幅と高さを取得
Declare Function Api_GetTextExtentPoint32& Lib "gdi32" Alias "GetTextExtentPoint32A"
(ByVal hDC&, ByVal lpsz$, ByVal cbString&, lpSize As POINTAPI)

' 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&)

' 印刷ジョブを終了
Declare Function Api_EndDoc& Lib "gdi32" Alias "EndDoc" (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&)

' 論理ペンを作成
Declare Function Api_CreatePen& Lib "gdi32" Alias "CreatePen" (ByVal nPenStyle&, ByVal
nWidth&, ByVal crColor&)

#define DI_COMPAT 4                          'システムイメージで描画する
#define DI_DEFAULTSIZE 8                     'widthとheightが0である場合、アイコン(マウスカーソ
                                             ル)をデフォルトのサイズで描画
#define DI_IMAGE 2                           'イメージを使ってアイコン(またはマウスカーソル)を描画
#define DI_MASK 1                            'マスクを使ってアイコン(またはマウスカーソル)を描画
#define DI_NORMAL 3                          'DI_IMAGE と DI_MASK の組み合わせ

#define PHYSICALHEIGHT 111                   '物理的高さ(単位ピクセル)
#define PHYSICALOFFSETX 112                  '実際に印刷可能なX方向のマージン
#define PHYSICALOFFSETY 113                  '実際に印刷可能なY方向のマージン
#define PHYSICALWIDTH 110                    '物理的幅(単位ピクセル)

#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_SOLID 0                           '実線のペンを作成

#define PRINTER_ENUM_DEFAULT &H1             'デフォルトのプリンタに関する情報を列挙
#define PRINTER_ENUM_LOCAL &H2               'Nameの設定を無視して、ローカルプリンタを列挙
#define PRINTER_ENUM_NAME &H8                'Nameで指定されたプリンタを列挙
#define PRINTER_ENUM_SHARED &H20             '共有属性を持つプリンタを列挙
#define MAX_DEVICENAME 64

Var Shared Combo1 As Object
Var Shared Button1 As Object
Var Shared Radio(4) As Object
Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
```

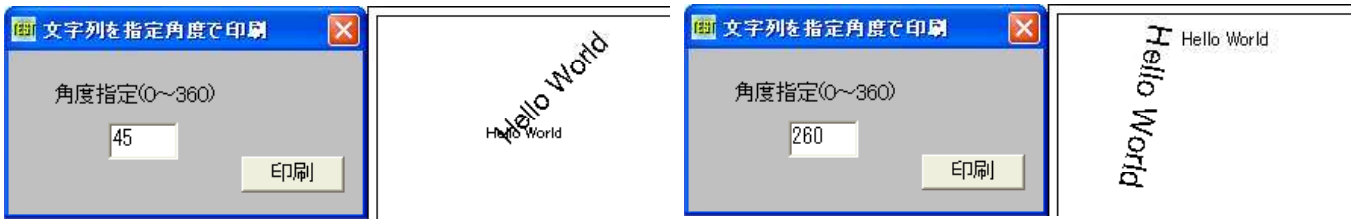
```
For i = 0 To 4
    Radio(i).Attach GetDlgItem("Radio" & Trim$(Str$(i + 1))) : Radio(i).SetFontSize 14
Next i

Var Shared PrinterName As String

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

'=====================================
'= 接続プリンタを列挙
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var PrintServer As String
    Var Needed As Long
    Var Returned As Long
    Var Level As Long
    Var CT As Long
    Var Ret As Long

    'ローカルプリンタから検索
    PrintServer = ""

    'PRINTER_INFO_5構造体を受け取る
    Level = 5

    'バッファに必要なバイト数を調べる
    Ret = Api_EnumPrinters(PRINTER_ENUM_NAME, PrintServer, Level, Chr$(0), 0, Needed,
Returned)
    If Needed = 0 Then End

    '全プリンタ情報を得る
    Var Buffer(Needed-1) As Byte
    Ret = Api_EnumPrinters(PRINTER_ENUM_NAME, PrintServer, Level, Buffer(0), Needed,
Needed, Returned)

    '構造体リストの準備
    Var PI_5(Returned - 1) As PRINTER_INFO_5

    For CT = 0 To Returned - 1

        'バッファから構造体1つ分を抜き取る
        CopyMemory PI_5(CT), Buffer(CT * Len(PI_5(CT))), Len(PI_5(CT))

        'プリンタ名を得る
        PrinterName = String$(MAX_DEVICENAME, Chr$(0))
        CopyMemory PrinterName, ByVal PI_5(CT).pPrinterName, Len(PrinterName)
        PrinterName = KLeft$(PrinterName, kInStr(1, PrinterName, Chr$(0)) - 1)

        Combo1.AddString PrinterName
    Next
End Sub

'=====================================
'= デフォルトプリンタの設定
'=====================================
Declare Sub Combo1_Change edecl ()
Sub Combo1_Change()
    Var hPrinter As Long
    Var Ret As Long

    PrinterName = Combo1.GetText(Combo1.GetCursel)
    Ret = Api_SetDefaultPrinter(PrinterName)
End Sub
```

```
'====================================
'= 印刷（文字列と、それを囲む楕円）
'====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var di As DOCINFO
    Var pa As POINTAPI
    Var PrinterDC As Long
    Var hPen As Long
    Var hOldPen As Long
    Var OutStr As String
    Var tLeft As Long
    Var tTop As Long
    Var tWidth As Long
    Var tHeight As Long
    Var pWidth As Long
    Var pHeight As Long
    Var pxMargin As Long
    Var pyMargin As Long
    Var Ret As Long

    If Combo1.GetWindowText = "" Then Exit Sub

    'プリンタのデバイスコンテキストを取得
    PrinterDC = Api_CreateDC(ByVal 0, PrinterName, Chr$(0), ByVal 0)
    If PrinterDC = 0 Then Exit Sub

    OutStr = "FB Programming Tips" & Chr$(0)

    pWidth = Api_GetDeviceCaps(PrinterDC, PHYSICALWIDTH)
    pHeight = Api_GetDeviceCaps(PrinterDC, PHYSICALHEIGHT)
    pyMargin = Api_GetDeviceCaps(PrinterDC, PHYSICALOFFSETY)
    pxMargin = Api_GetDeviceCaps(PrinterDC, PHYSICALOFFSETX)

    di.cbSize = Len(di)
    di.lpszdiName = StrAdr(OutStr)

    Ret = Api_StartDoc(PrinterDC, di)
    Ret = Api_StartPage(PrinterDC)

    hPen = Api_CreatePen(Index, 0, RGB(255, 0, 0))
    hOldPen = Api_SelectObject(PrinterDC, hPen)

    '文字サイズを取得
    Ret = Api_GetTextExtentPoint32(PrinterDC, OutStr, Len(OutStr), pa)

    tWidth = pa.X
    tHeight = pa.Y
    tTop = (pHeight - (2 * pyMargin) - pa.Y) / 2
    tLeft = (pWidth - (2 * pxMargin) - pa.X) / 2

    '文字を囲む楕円を描画
    Ret = Api_Ellipse(PrinterDC, tLeft - 50, tTop - 50, tLeft + 50 + tWidth, tTop + 50 +
tHeight)

    '文字列を描画
    OutStr = Left$(OutStr, Instr(OutStr, Chr$(0)) - 1)
    Ret = Api_TextOut(PrinterDC, tLeft, tTop, OutStr, Len(OutStr))

    Ret = Api_EndPage(PrinterDC)
    Ret = Api_EndDoc(PrinterDC)
    Ret = Api_DeleteObject(hPen)
    Ret = Api_DeleteDC(PrinterDC)
End Sub

'====================================
'=
'====================================
While 1
    WaitEvent
```

```
Wend
Stop
End
```

APIを使った印刷(文字列を指定角度で)

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
DeleteObject 論理オブジェクトを削除し、システムリソースを解放
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
TextOut 文字を描画
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




```
'=====================================
'= APIを使った印刷(文字列を指定角度で)
'=    (PrintRotatedText.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 As String * LF_FACESIZE
End Type

Type DOCINFO
    cbSize            As Long
    lpszDocName       As Long
    lpszOutput        As Long
    lpszDatatype      As Long
    fwType            As Long
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&)
```

```
' 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, ByVal 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&)

' 文字を描画
Declare Function Api_TextOut& Lib "gdi32" Alias "TextOutA" (ByVal hDC&, ByVal nXStart&,
ByVal nYStart&, ByVal lpString$, ByVal cbString&)

' 印刷ジョブを開始
Declare Function Api_StartDoc& Lib "gdi32" Alias "StartDocA" (ByVal hDC&, lpdi As
DOCINFO)

' 印刷ジョブを終了
Declare Function Api_EndDoc& Lib "gdi32" Alias "EndDoc" (ByVal hDC&)

' 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&)

#define DESIREDFONTSIZE 12

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 OutString As String                 '回転文字列
    Var lf As LOGFONT                       '回転フォントの構造体
    Var hOldfont As Long                    '旧フォント情報を待避
    Var hPrintDc As Long                    'プリンタのデバイスコンテキスト
    Var hFont As Long                       '新規フォントのハンドル
    Var di As DOCINFO                       'ドキュメント情報の構造体
    Var Ret As Long                         '戻り値

    '回転させる文字列
    OutString = "Hello World"

    '文字サイズ・回転角度を設定
    lf.lfEscapement = Val(Edit1.GetWindowText) * 10
    lf.lfHeight = (DESIREDFONTSIZE * 16)

    '回転文字フォントを作成
    hFont = Api_CreateFontIndirect(lf)
    di.cbSize = Len(di)
    di.lpszDocName = StrAdr("My Document" & Chr$(0))

    'デフォルトプリンタのデバイスコンテキストを作成
    hPrintDc = Api_CreateDC(ByVal 0, "FinePrint", 0, 0)

    '印刷開始処理
    Ret = Api_StartDoc(hPrintDc, di)
    Ret = Api_StartPage(hPrintDc)
```

```
    'プリンタデバイスコンテキストを選択
    hOldfont = Api_SelectObject(hPrintDc, hFont)

    '回転テキストを開始位置(1000, 1000)に送る
    Ret = Api_TextOut(hPrintDc, 1000, 1000, OutString, Len(OutString))

    '以前のオブジェクトに戻る
    Ret = Api_SelectObject(hPrintDc, hOldfont)

    '通常のテキストを描画
    Ret = Api_TextOut(hPrintDc, 1000, 1000, OutString, Len(OutString))

    '印刷終了処理
    Ret = Api_EndPage(hPrintDc)
    Ret = Api_EndDoc(hPrintDc)

    '解放処理
    Ret = Api_DeleteDC(hPrintDc)
    Ret = Api_DeleteObject(hFont)
End Sub

'=====================================
'= 文字入力 + Enter
'=====================================
Declare Sub Edit1_Change edecl ()
Sub Edit1_Change()
    Var EPos As Integer

    Ed$ = Edit1.GetWindowText

    EPos = InStr(Ed$, Chr$(13, 10))
    If EPos <> 0 Then
        Ed$ = Mid$(Ed$, 1, EPos - 1) & Mid$(Ed$, EPos + 2)
        Edit1.SetWindowText Ed$
        If Val(Ed$) < 0 Or Val(Ed$) > 360 Then
            Edit1.SetWindowText ""
            Edit1.SetFocus
        Else
            Button1.SetFocus
        End If
    End If
End Sub

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

AVIファイルの再生(Ⅰ)

AVIファイルを再生します。
mciSendString 文字列を MCI に送信
mciGetErrorString MCIエラーコードを記述する文字列を取得
GetShortPathName ファイルの短い形式のパス名を取得

[…] ボタンでAVIファイルを選択し、[▶] ボタンで再生します。

```
'=====================================
'= AVIファイルの再生
'=    (mciSendString.bas)
'=====================================
#include "Windows.bi"

' 文字列を MCI に送信
Declare Function Api_mciSendString& Lib "winmm" Alias "mciSendStringA" (ByVal
lpstrCommand$, ByVal lpstrReturnString$, ByVal uReturnLength&, ByVal hwndCallback&)

' MCIエラーコードを記述する文字列を取得
Declare Function Api_mciGetErrorString& Lib "winmm" Alias "mciGetErrorStringA" (ByVal
dwError&, ByVal lpstrBuffer$, ByVal uLength&)

' 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 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

Var Shared Text1 As Object
Var Shared Button1 As Object
Var Shared Button2 As Object
Var Shared Picture1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 12
Button1.Attach GetDlgItem("Button1") : button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : button2.SetFontSize 14
Picture1.Attach GetDlgItem("Picture1")

Var Shared FileName As String

'=====================================
'= […]
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()

    FileName = WinOpenDlg("ファイルのオープン","C:¥*.avi","全てのﾌｧｲﾙ(*.avi)", 0)
    If FileName <> Chr$(&H1B) Then
        Text1.SetWindowText FileName
    End If
End Sub

'=====================================
'= [▶]
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var CommandString As String
    Var ShortFileName As String * 260
    Var deviceIsOpen As Integer
    Var Ret As Long

    FileName = GetDlgItemText("Text1")
```

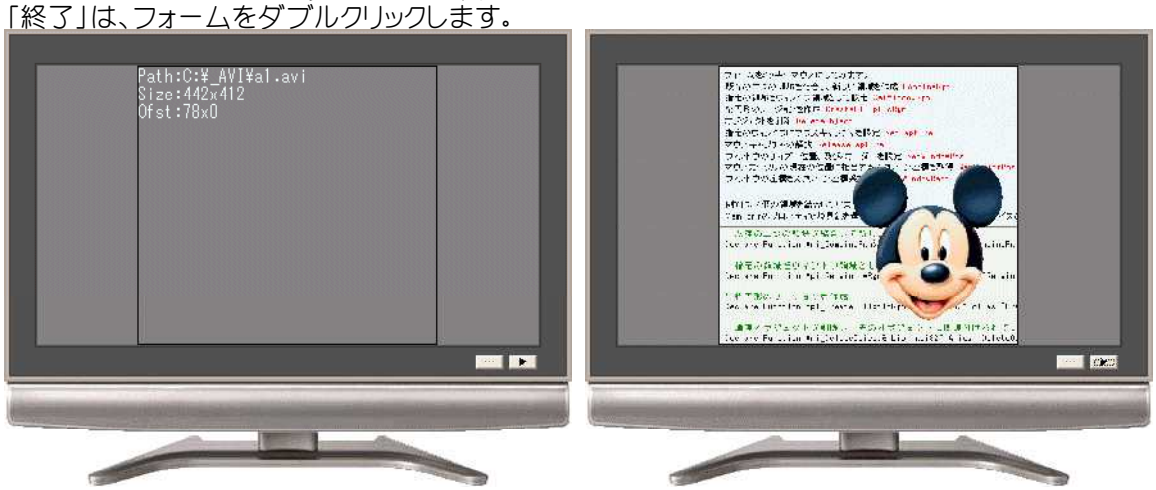
```
    Ret = Api_GetShortPathName(FileName, ShortFileName, Len(ShortFileName))
    FileName = Left$(ShortFileName, Ret)

    'デバイスオープン
    CommandString = "Open " & FileName & " Type AVIVideo Alias AVIFile parent " &
Str$(Picture1.GethWnd) & " style " & Str$(WS_CHILD)
    Ret = Api_mciSendString(CommandString, ByVal 0, 0, 0)
    If Ret Then goto *Err_Trap

    deviceIsOpen = True

    'ピクチャボックスサイズに合わせる
    CommandString = "put AVIFile window at 0 0 " & Str$(Picture1.GetWidth ) & " " &
Str$(Picture1.GetHeight)
    Ret = Api_mciSendString(CommandString, ByVal 0, 0, 0)
    If Ret <> 0 Then goto *Err_Trap

    'ファイル再生
    CommandString = "Play AVIFile wait"
    Ret = Api_mciSendString(CommandString, ByVal 0, 0, 0)
    If Ret <> 0 Then goto *Err_Trap

    'デバイスクローズ
    CommandString = "Close AVIFile"
    Ret = Api_mciSendString(CommandString, ByVal 0, 0, 0)
    If Ret <> 0 Then goto *Err_Trap

    Exit Sub

*Err_Trap
    Var ErrorString As String
    ErrorString = space$(256)
    Ret = Api_mciGetErrorString(Ret, ErrorString, Len(ErrorString))
    ErrorString = Left$(ErrorString, InStr(ErrorString, Chr$(0)) - 1)

    If deviceIsOpen Then
        CommandString = "Close AVIFile"
        Ret = Api_mciSendString(CommandString, ByVal 0, 0, 0)
    End If
End Sub

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

AVIファイルの再生(II)

AVIファイルを再生(I)と同じですが、透過処理でフォームをCTV風にしてみました。
AVIFileOpen AVIファイルオープン
AVIFileRelease AVIファイルハンドル解放
AVIFileInfo AVIファイル情報取得
AVIFileInit AVIファイル初期化
AVIFileExit AVIファイルライブラリ解放
mciSendString 文字列をMCIに送信
mciGetErrorString MCIエラーコードを記述する文字列を取得
GetShortPathName ファイルの短い形式のパス名を取得
GetCursorPos マウスポインタのスクリーン座標を取得
SetWindowPos ウィンドウのサイズ、位置、およびZオーダーを設定
GetWindowRect ウィンドウの座標をスクリーン座標系で取得
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
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の情報を取得
SetLayeredWindowAttributes 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

Picture1のサイズ(391x218)より大きなAVIファイルは、その比率を維持し縮小、小さなAVIファイルは、実サイズで再生します。
図のBMPファイルをフォームに貼り付け、スタンド部の背景色RGB(0, 128, 128)をキーとして透明にしています。
「…」でファイルのオープンダイアログを開く、またはAVIファイルをフォーム上にドラッグ＆ドロップしてください。
「終了」は、フォームをダブルクリックします。




```
'=======================================
'= AVIファイルの再生(II)
'=    (mciSendString3.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

Type AVIFILEINFO
    dwMaxBytesPerSec As Long              'ファイルのデータレートのほぼ最大値
    dwFlags          As Long              '拡張可能なフラグ
    dwCaps           As Long              '適応フラグ
    dwStreams        As Long              'ファイル中のストリーム数
    dwSuggestedBufferSize As Long         '読み込み時に必要となる予想されるバッファサイズ(バイト)
    dwWidth          As Long              'AVIファイル中の幅(ピクセル)
    dwHeight         As Long              'AVIファイル中の高さ(ピクセル)
    dwScale          As Long              '全ファイルに適応できるタイムスケール
    dwRate           As Long              '(dwRate÷dwScale)は秒間サンプル数
    dwLength         As Long              'AVIファイルサイズ。単位は(dwRate÷dwScale)
    dwEditCount      As Long              'AVIファイルに追加、またはAVIファイルから削除されたストリーム数
    szFileType       As String * 64       'ファイルタイプ情報の記述を含む、Nullで終わる文字列
End Type

' AVIファイルオープン
Declare Function Api_AVIFileOpen& Lib "avifil32" Alias "AVIFileOpenA" (ppFile&, ByVal
szFile$, ByVal Mode&, pclsidHandler As Any)

' AVIファイルハンドル解放
Declare Function Api_AVIFileRelease& Lib "avifil32" Alias "AVIFileRelease" (ByVal
pFile&)

' AVIファイル情報取得
Declare Function Api_AVIFileInfo& Lib "avifil32" Alias "AVIFileInfoA" (ByVal pFile&, pfi
```

```
As AVIFILEINFO, ByVal lSize&)

' AVIファイル初期化
Declare Sub Api_AVIFileInit Lib "avifil32" Alias "AVIFileInit" ()

' AVIファイルライブラリ解放
Declare Sub Api_AVIFileExit Lib "avifil32" Alias "AVIFileExit" ()

' 文字列を MCI に送信
Declare Function Api_mciSendString& Lib "winmm" Alias "mciSendStringA" (ByVal
lpstrCommand$, ByVal lpstrReturnString As Any, ByVal uReturnLength&, ByVal
hwndCallback&)

' MCIエラーコードを記述する文字列を取得
Declare Function Api_mciGetErrorString& Lib "winmm" Alias "mciGetErrorStringA" (ByVal
dwError&, ByVal lpstrBuffer$, ByVal uLength&)

' 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&)

' 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)

' 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)

' 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 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 PS_SOLID 0
#define RGN_AND 1                          'リージョン同士のAND結合
#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 WS_EX_LAYERED &H80000              '透明なウィンドウ属性(Windows2000以上)
#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 GWL_EXSTYLE -20                    '拡張ウィンドウスタイル

#define OF_SHARE_DENY_WRITE &H20           '他のプロセスから書き込み可能

#define HWND_BOTTOM 1                      'ウィンドウを最背面に配置
#define HWND_NOTOPMOST (-2)                'ウィンドウを常に最前面に配置(他のウィンドウが
```

```
                                                HWND_TOPMOSTに配置されている場合はその配下)
#define HWND_TOP 0                              'ウィンドウを最前面に配置
#define HWND_TOPMOST (-1)                       'ウィンドウを常に最前面に配置

#define SWP_SHOWWINDOW &H40                     'ウインドウを表示する
#define SWP_NOZORDER &H4                        'ウィンドウリスト内で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_NOMOVE &H2                          'ウインドウの現在位置を保持する

Var Shared Button1 As Object
Var Shared Button2 As Object
Var Shared Picture1 As Object
Var Shared Bitmap As Object
BitmapObject Bitmap

Button1.Attach GetDlgItem("Button1") : button1.SetFontSize 9
Button2.Attach GetDlgItem("Button2") : button2.SetFontSize 9
Picture1.Attach GetDlgItem("Picture1")

Var Shared FileName As String
Var Shared StartCursor As POINTAPI
Var Shared StartPos As RECT
Var Shared Capture As Integer
Var Shared aviWidth As Integer
Var Shared aviHeight As Integer

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

    Bitmap.LoadFile "CTV.bmp"
    DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject
    Picture1.SetFontName "MS ゴシック"

    dwStyle = Api_GetWindowLong(GethWnd, GWL_EXSTYLE)
    dwStyle = dwStyle Or WS_EX_LAYERED
    Ret = Api_SetWindowLong(GethWnd, GWL_EXSTYLE, dwStyle)
    Ret = Api_SetLayeredWindowAttributes(GethWnd, RGB(0, 128, 128), 0, LWA_COLORKEY)
    Ret = Api_SetWindowPos(GethWnd, HWND_TOPMOST, 0, 0, 0, 0, SWP_SHOWWINDOW Or SWP_NOMOVE
Or SWP_NOSIZE)

    ShowWindow -1
End Sub

'=====================================
'=
'=====================================
Declare Sub PathAndSizeSet()
Sub PathAndSizeSet()
    Var hFile As Long
    Var AviInfo As AVIFILEINFO
    Var newPicWidth As Integer
    Var newPicHeight As Integer
    Var hOffset As Integer
    Var vOffset As Integer
    Var Ret As Long

    Picture1.MoveWindow 26, 28
    Picture1.SetWindowSize 391, 218
    Picture1.Cls

    'AVIファイル初期化
    Api_AVIFileInit
```

```
    'AVIファイルハンドル取得
    Ret = Api_AVIFileOpen(hFile, FileName, OF_SHARE_DENY_WRITE, ByVal 0)
    Ret = Api_AVIFileInfo(hFile, AviInfo, Len(AviInfo))
    aviWidth = AviInfo.dwWidth
    aviHeight = AviInfo.dwHeight

    'AVIサイズを縮小
    If aviWidth > Picture1.GetWidth Or aviHeight > Picture1.GetHeight Then

        '10:9より比率が大の場合
        If aviHeight / aviWidth > 0.56 Then
            newPicWidth = Picture1.GetHeight / aviHeight * aviWidth
            hOffset = (Picture1.GetWidth - newPicWidth) / 2
            Picture1.MoveWindow 26 + hOffset, 28
            Picture1.SetWindowSize newPicWidth, 218

        '10:9より比率が小の場合
        Else
            newPicHeight = Picture1.GetWidth / aviWidth * aviHeight
            vOffset = (Picture1.GetHeight - newPicHeight) / 2
            Picture1.MoveWindow 26, 28 + vOffset
            Picture1.SetWindowSize 391, newPicHeight
        End If

    'AVIサイズをそのまま
    Else
        hOffset = (391 - aviWidth) / 2
        vOffset = (218 - aviHeight) / 2
        Picture1.MoveWindow 26 + hOffset, 28 + vOffset
        Picture1.SetWindowSize aviWidth, aviHeight
    End If

    'AVIファイル名
    Picture1.Print "Path:" & FileName

    'AVIサイズ
    Picture1.Print "Size:" & Trim$(Str$(aviWidth)) & "x" & Trim$(Str$(aviHeight))
    Picture1.Print "Ofst:" & Trim$(Str$(hOffset)) & "x" & Trim$(Str$(vOffset))
End Sub

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    FileName = WinOpenDlg("ファイルのオープン","C:¥*.avi","AVIファイル(*.avi)", 0)
    If FileName <> Chr$(&H1B) Then
        PathAndSizeSet
    End If
End Sub

'=======================================
'=
'=======================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var CommandString As String
    Var ShortFileName As String * 260
    Var deviceIsOpen As Integer
    Var Ret As Long

    Ret = Api_GetShortPathName(FileName, ShortFileName, Len(ShortFileName))
    FileName = Left$(ShortFileName, Ret)

    Ret = Api_mciSendString("Close All", ByVal 0, 0, 0)

    'デバイスオープン
    Ret = Api_mciSendString("Open " & FileName & " Type AVIVideo Alias Video Parent " &
Str$(Picture1.GethWnd) & " Style " & Str$(WS_CHILD), ByVal 0, 0, 0)
    If Ret Then Goto *Err_Trap
```

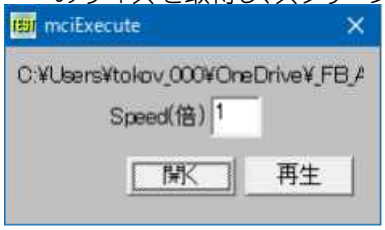
```
    deviceIsOpen = True

    'ピクチャボックスサイズに合わせる
    Ret = Api_mciSendString("Put Video Window at 0 0 " & Str$(Picture1.GetWidth ) & " " &
Str$(Picture1.GetHeight), ByVal 0, 0, 0)
    If Ret <> 0 Then Goto *Err_Trap

    'ファイル再生
    Ret = Api_mciSendString("Play Video Wait", ByVal 0, 0, 0)
    If Ret <> 0 Then Goto *Err_Trap

    'デバイスクローズ
    Ret = Api_mciSendString("Close Video", ByVal 0, 0, 0)
    If Ret <> 0 Then Goto *Err_Trap
    Exit Sub

*Err_Trap
    Var ErrorString As String
    ErrorString = Space$(256)
    Ret = Api_mciGetErrorString(Ret, ErrorString, Len(ErrorString))
    ErrorString = Left$(ErrorString, InStr(ErrorString, Chr$(0)) - 1)

    If deviceIsOpen Then
        Ret = Api_mciSendString("Close Video", ByVal 0, 0, 0)
    End If
End Sub

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

    CN = GetDropFileCount(DF)
    FileName = GetDropFileName(DF, 0)
    PathAndSizeSet
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)
    Var Ret As Long

    If Button = 1 Then

        'ドラッグ
        Capture = True
        Ret = Api_GetCursorPos(StartCursor)
        Ret = Api_GetWindowRect(GethWnd, StartPos)
        Ret = Api_SetCapture(GethWnd)
    End If
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 CurrentCursor As POINTAPI
    static Event As Integer
    Var Ret As Long

    If Capture And Button = 1 And Not Event Then
```

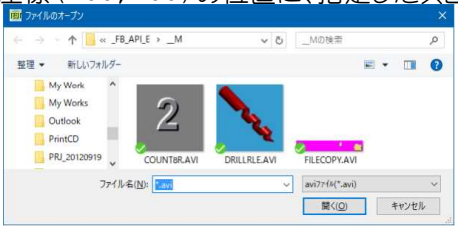
```
        Event = True
        CallEvent

        Ret = Api_GetCursorPos(CurrentCursor)

        Ret = Api_SetWindowPos(GethWnd, 0, StartPos.Left + CurrentCursor.x -
StartCursor.x, StartPos.Top + CurrentCursor.y - StartCursor.y, 0, 0, SWP_NOZORDER Or
SWP_NOSIZE)
        Event = False
    End If
End Sub

'=======================================
'=
'=======================================
Declare Sub MainForm_MouseUp edecl (ByVal Button As Integer, ByVal Shift As Integer, ByVal
x As Single, ByVal y As Single)
Sub MainForm_MouseUp(ByVal Button As Integer, ByVal Shift As Integer, ByVal x As Single,
ByVal y As Single)
    Var Ret As Long

    If Button = 1 And Capture Then

        'ドラッグ停止
        Capture = False
        Ret = Api_ReleaseCapture
    End If
End Sub

'=======================================
'=
'=======================================
Declare Sub MainForm_DblClick edecl ()
Sub MainForm_DblClick()
    End
End Sub

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

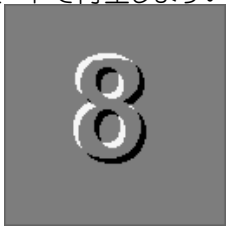
AVIファイルの再生(III)

AVIFileOpen AVIファイルオープン
AVIFileRelease AVIファイルハンドル解放
AVIFileInfo AVIファイル情報取得
AVIFileInit AVIファイル初期化
AVIFileExit AVIファイルライブラリ解放
mciSendString 文字列をMCIに送信
mciGetErrorString MCIエラーコードを記述する文字列を取得
mciExecute MCIにアクセス

AVIのサイズを取得し、スクリーン座標(100,100)の位置に、指定したスピードで再生します。







```
'=====================================
'= AVIファイルの再生(Ⅲ)
'=    (mciExecute.bas)
'=====================================
#include "Windows.bi"

Type AVIFILEINFO
    dwMaxBytesPerSec As Long                'ファイルのデータレートのほぼ最大値
    dwFlags          As Long                '拡張可能なフラグ
    dwCaps           As Long                '適応フラグ
    dwStreams        As Long                'ファイル中のストリーム数
    dwSuggestedBufferSize As Long           '読み込み時に必要となる予想されるバッファサイズ(バイト)
    dwWidth          As Long                'AVIファイル中の幅(ピクセル)
    dwHeight         As Long                'AVIファイル中の高さ(ピクセル)
    dwScale          As Long                '全ファイルに適応できるタイムスケール
    dwRate           As Long                '(dwRate÷dwScale)は秒間サンプル数
    dwLength         As Long                'AVIファイルサイズ。単位は(dwRate÷dwScale)
    dwEditCount      As Long                'AVIファイルに追加、またはAVIファイルから削除されたストリーム数
    szFileType       As String * 64         'ファイルタイプ情報の記述を含む、Nullで終わる文字列
End Type

' AVIファイルオープン
Declare Function Api_AVIFileOpen& Lib "avifil32" Alias "AVIFileOpenA" (ppFile&, ByVal
szFile$, ByVal Mode&, pclsidHandler As Any)

' AVIファイルハンドル解放
Declare Function Api_AVIFileRelease& Lib "avifil32" Alias "AVIFileRelease" (ByVal
pFile&)

' AVIファイル情報取得
Declare Function Api_AVIFileInfo& Lib "avifil32" Alias "AVIFileInfoA" (ByVal pFile&, pfi
As AVIFILEINFO, ByVal lSize&)

' AVIファイル初期化
Declare Sub Api_AVIFileInit Lib "avifil32" Alias "AVIFileInit" ()

' AVIファイルライブラリ解放
Declare Sub Api_AVIFileExit Lib "avifil32" Alias "AVIFileExit" ()

' MCIにアクセス
Declare Function Api_mciExecute& Lib "winmm" Alias "mciExecute" (ByVal lpstrCommand$)

#define OF_SHARE_DENY_WRITE &H20            '他のプロセスから書き込み可能

Var Shared sAVIFile As String
Var Shared AviInfo As AVIFILEINFO
Var Shared aviWidth As Long
Var Shared aviHeight As Long

Var Shared Text1 As Object
Var Shared Edit1 As Object

Text1.Attach GetDlgItem("Text1")
Edit1.Attach GetDlgItem("Edit1")

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hFile As Long

    sAVIFile = WinOpEndlg("ファイルのオープン", "*.avi", "aviﾌｧｲﾙ(*.avi)", 0)

    If sAVIFile <> Chr$(&H1B) Then
        Text1.SetWindowText sAVIFile
```

```
        'AVIファイル初期化
        Api_AVIFileInit

        'AVIファイルハンドル取得
        Ret = Api_AVIFileOpen(hFile, sAVIFile, OF_SHARE_DENY_WRITE, ByVal 0)

        If Ret = 0 Then

            'AVIインフォメーション検索
            Ret = Api_AVIFileInfo(hFile, AviInfo, Len(AviInfo))

            'AVIファイルのサイズを取得
            If Ret = 0 Then
                aviWidth = AviInfo.dwWidth
                aviHeight = AviInfo.dwHeight
            End If

            'AVIファイルハンドル解放
            Ret = Api_AVIFileRelease(hFile)
        Else
            A% = MessageBox("", "AVIファイルオープンエラー", 0, 2)
        End If

        Api_AVIFileExit
    Else
        sAVIFile = ""
        Text1.SetWindowText sAVIFile
    End If

End Sub

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

    If sAVIFILE = "" Then Exit Sub

    Spd = Val(Edit1.GetWindowText)
    If Spd < 1 Or Spd > 10 Then Spd = 5 : Edit1.SetWindowText Trim$(Str$(Spd))

    'AVIファイルを開く
    Ret = Api_mciExecute("Open " & sAVIFile & " Type AviVideo Alias tokovideo Style Popup")

    'ディスプレイウィンドウを(100,100)の位置に設定
    Ret = Api_mciExecute("Put tokovideo Window At 100 100 " & Trim$(Str$(aviWidth)) & " " &
Trim$(Str$(aviHeight)))

    '再生スピードをSpd倍に
    Ret = Api_mciExecute("Set tokovideo Speed " & Trim$(Str$(Spd * 1000)))

    'AVIの再生
    Ret = Api_mciExecute("Play tokovideo Wait")

    'メモリークローズ
    Ret = Api_mciExecute("Close tokovideo")
End Sub

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

AVIファイルの情報取得(Ⅰ)

AVIファイルの情報を取得します。
AVIFileOpen AVIファイルオープン
AVIFileInfo AVIファイル情報取得
AVIFileRelease AVIファイルハンドル解放
AVIFileInit AVIファイル初期化
AVIFileExit AVIファイルライブラリ解放

エディットボックスにAVIファイルをドロップダウン入力し、『Info取得』をクリックし、情報を取得します。

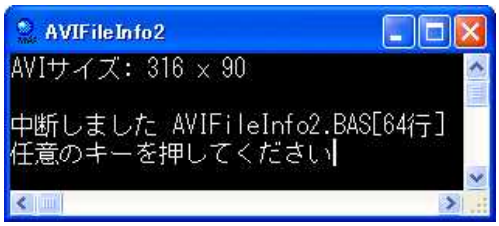

```
'=======================================
'= AVIファイル情報取得
'=    (AVIFileInfo.bas)
'=======================================
#include "Windows.bi"

Type AVIFILEINFO
    dwMaxBytesPerSec As Long                'ファイルのデータレートのほぼ最大値
    dwFlags          As Long                '拡張可能なフラグ
    dwCaps           As Long                '適応フラグ
    dwStreams        As Long                'ファイル中のストリーム数
    dwSuggestedBufferSize As Long           '読み込み時に必要となる予想されるバッファサイズ(バイト)
    dwWidth          As Long                'AVIファイル中の幅(ピクセル)
    dwHeight         As Long                'AVIファイル中の高さ(ピクセル)
    dwScale          As Long                '全ファイルに適応できるタイムスケール
    dwRate           As Long                '(dwRate÷dwScale)は秒間サンプル数
    dwLength         As Long                'AVIファイルサイズ。単位は(dwRate÷dwScale)
    dwEditCount      As Long                'AVIファイルに追加、又はAVIファイルから削除されたストリーム数
    szFileType       As String * 64         'ファイルタイプ情報の記述を含む、Nullで終わる文字列
End Type

' AVIファイルオープン
Declare Function Api_AVIFileOpen& Lib "avifil32" Alias "AVIFileOpenA" (ppFile&, ByVal
szFile$, ByVal Mode&, pclsidHandler As Any)

' AVIファイルハンドル解放
Declare Function Api_AVIFileRelease& Lib "avifil32" Alias "AVIFileRelease" (ByVal
pFile&)

' AVIファイル情報取得
Declare Function Api_AVIFileInfo& Lib "avifil32" Alias "AVIFileInfoA" (ByVal pFile&, pfi
As AVIFILEINFO, ByVal lSize&)

' AVIファイル初期化
Declare Sub Api_AVIFileInit Lib "avifil32" Alias "AVIFileInit" ()

' AVIファイルライブラリ解放
Declare Sub Api_AVIFileExit Lib "avifil32" Alias "AVIFileExit" ()

#define OF_SHARE_DENY_WRITE &H20            '他のプロセスから書き込み可能
```

```
Var Shared Text1 As Object
Var Shared Edit1 As Object
Var Shared List1 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var Shared FileName As String

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hFile As Long
    Var afi As AVIFILEINFO
    Var Ret As Long

    FileName = Edit1.GetWindowText

    'AVIファイル初期化
    Api_AVIFileInit

    'AVIファイルハンドル取得
    Ret = Api_AVIFileOpen(hFile, FileName, OF_SHARE_DENY_WRITE, ByVal 0)

    If Ret = 0 Then

        'AVIインフォメーション検索
        Ret = Api_AVIFileInfo(hFile, afi, Len(afi))

        If Ret = 0 Then
            List1.ResetContent
            List1.AddString "dwMaxBytesPerSec     =" & Str$(afi.dwMaxBytesPerSec)
            List1.AddString "dwFlags              =" & Str$(afi.dwFlags)
            List1.AddString "dwCaps               =" & Str$(afi.dwCaps)
            List1.AddString "dwStreams            =" & Str$(afi.dwStreams)
            List1.AddString "dwSuggestedBufferSize=" & Str$(afi.dwSuggestedBufferSize)
            List1.AddString "dwWidth              =" & Str$(afi.dwWidth)
            List1.AddString "dwHeight             =" & Str$(afi.dwHeight)
            List1.AddString "dwScale              =" & Str$(afi.dwScale)
            List1.AddString "dwRate               =" & Str$(afi.dwRate)
            List1.AddString "dwLength             =" & Str$(afi.dwLength)
            List1.AddString "dwEditCount          =" & Str$(afi.dwEditCount)
            List1.AddString "szFileType           =" & afi.szFileType
        Else
            A% = MessageBox("", "AVIファイルオープンエラー", 0, 2)
        End If

        'AVIファイルハンドル解放
        Ret = Api_AVIFileRelease(hFile)
    Else
        A% = MessageBox("", "AVIファイルオープンエラー", 0, 2)
    End If

    Api_AVIFileExit
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
```

AVIファイルの情報取得(II)

AVIファイルの情報からサイズを取得します。
AVIFileOpen AVIファイルオープン
AVIFileInfo AVIファイル情報取得
AVIFileRelease AVIファイルハンドル解放
AVIFileInit AVIファイル初期化
AVIFileExit AVIファイルライブラリ解放

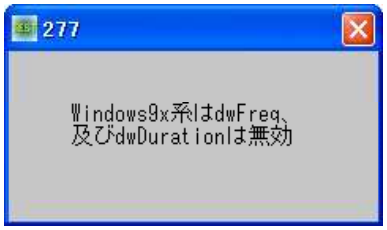



```
'=====================================
'= AVIファイルの情報取得(II)
'=    (AVIFileInfo2.bas)
'=====================================
#define OF_SHARE_DENY_WRITE &H20

Type AVIFILEINFO
    dwMaxBytesPerSec As Long                'ファイルのデータレートのほぼ最大値
    dwFlags          As Long                '拡張可能なフラグ
    dwCaps           As Long                '適応フラグ
    dwStreams        As Long                'ファイル中のストリーム数
    dwSuggestedBufferSize As Long           '読み込み時に必要となる予想されるバッファサイズ(バイト)

    dwWidth          As Long                'AVIファイル中の幅(ピクセル)
    dwHeight         As Long                'AVIファイル中の高さ(ピクセル)
    dwScale          As Long                '全ファイルに適応できるタイムスケール
    dwRate           As Long                '(dwRate÷dwScale)は秒間サンプル数
    dwLength         As Long                'AVIファイルサイズ。単位は(dwRate÷dwScale)
    dwEditCount      As Long                'AVIファイルに追加、又はAVIファイルから削除されたストリーム数

    szFileType       As String * 64         'ファイルタイプ情報の記述を含む、Nullで終わる文字列
End Type

' AVIファイルオープン
Declare Function Api_AVIFileOpen& Lib "avifil32" Alias "AVIFileOpenA" (ppFile&, ByVal
szFile$, ByVal Mode&, pclsidHandler As Any)

' AVIファイルハンドル解放
Declare Function Api_AVIFileRelease& Lib "avifil32" Alias "AVIFileRelease" (ByVal
pFile&)

' AVIファイル情報取得
Declare Function Api_AVIFileInfo& Lib "avifil32" Alias "AVIFileInfoA" (ByVal pFile&, pfi
As AVIFILEINFO, ByVal lSize&)

' AVIファイル初期化
Declare Sub Api_AVIFileInit Lib "avifil32" Alias "AVIFileInit" ()
```

```
' AVIファイルライブラリ解放
Declare Sub Api_AVIFileExit Lib "avifil32" Alias "AVIFileExit" ()

'------------------------------
Var hFile As Long
Var FileName As String
Var AviInfo As AVIFileInfo
Var Ret As Long

FileName = "C:¥hoge¥hoge.avi"

Api_AVIFileInit

'AVIファイルのハンドル取得
If Api_AVIFileOpen(hFile, FileName, OF_SHARE_DENY_WRITE, ByVal 0) = 0 Then

    'AVIファイル情報取得
    If Api_AVIFileInfo(hFile, AviInfo, len(AviInfo)) = 0 Then
        Print "AVIサイズ: " & Trim$(Str$(AviInfo.dwWidth)) & " x " &
Trim$(Str$(AviInfo.dwHeight))
    Else
        Print "AVIファイル情報取得に失敗しました！ :("
    End If

    'AVIファイルのハンドル解放
    Ret = Api_AVIFileRelease(hFile)
Else
    Print "AVIファイル情報取得に失敗しました！ :("
End If

Api_AVIFileExit

Stop
End
```

BEEP音を発生

BEEP音を発生させます。Windows9x系ではdwFreq(周波数)、dwDuration(持続時間)は共に無効です。F-BASICでのBEEPに同じ。
WindowsNT系では、周波数及び持続時間を設定することができます。
Beep BEEP音を発生



テストでは、37Hzから997Hzまで50mSecの持続時間で可変させています。そのときの周波数をキャプションに表示させています。
ついでにドレミを…

```
'========================================
'=  Beep音を発生
'=    (Beep.bas)
'========================================
#include "Windows.bi"

' Beep音の発生
Declare Function Api_Beep& Lib "kernel32" Alias "Beep" (ByVal dwFreq&, ByVal dwDuration&)

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

```
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var Hz As Long                          'ヘルツ
    Var Ret As Long

    For Hz = 37 To 997 step 10              '指定可能範囲:37 To 32,767
        Ret = Api_Beep(Hz, 50)              '持続時間:50mmSec
        SetWindowText Trim$(Str$(Hz))
        CallEvent
    Next

    Ret = Api_Beep(262, 500)                'ド
    Ret = Api_Beep(294, 500)                'レ
    Ret = Api_Beep(330, 500)                'ミ
    Ret = Api_Beep(349, 500)                'ファ
    Ret = Api_Beep(392, 500)                'ソ
    Ret = Api_Beep(440, 500)                'ラ
    Ret = Api_Beep(494, 500)                'シ
    Ret = Api_Beep(523, 500)                'ド
End Sub

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

CallEventを独自に作成

PeekMessage メッセージキューのメッセージを取得
TranslateMessage 仮想キーのメッセージを文字メッセージに変換
DispatchMessage 構造体のメッセージを プロシージャに送出
PM_REMOVE(&H1) メッセージをキューから削除

例では、画面キャプチャのためCounter値、およびFor～Next内にWaitを入れています。

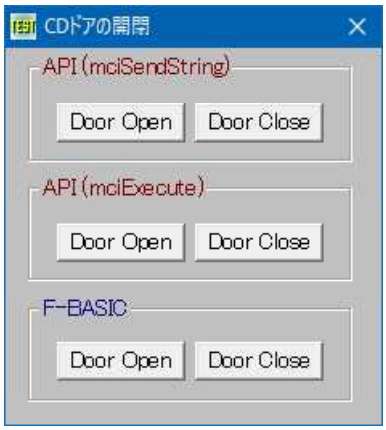

```
'=======================================
'= CallEventを独自に作成
'=    (OrgCallEvent.bas)
'=======================================
#include "Windows.bi"

Type POINTAPI
    x   As Long
    y   As Long
End Type

Type MSG
    hwnd        As Long
    message     As Long
    wParam      As Long
    lParam      As Long
    mtime       As Long
    pt          As POINTAPI
End Type
```

```
' メッセージキューのメッセージを取得
Declare Function Api_PeekMessage& Lib "user32" Alias "PeekMessageA" (lpMsg As Msg, ByVal
hWnd&, ByVal wMsgFilterMin&, ByVal wMsgFilterMax&, ByVal wRemoveMsg&)

' 仮想キーのメッセージを文字メッセージに変換
Declare Function Api_TranslateMessage& Lib "user32" Alias "TranslateMessage" (lpMsg As
MSG)

' 構造体のメッセージを プロシージャに送出
Declare Function Api_DispatchMessage& Lib "user32" Alias "DispatchMessageA" (lpMsg As
MSG)

#define PM_REMOVE &H1                          'メッセージをキューから削除

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

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 20
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 i

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

'=====================================
'=
'=====================================
Declare Sub OrgCallEvent edecl ()
Sub OrgCallEvent()
    Var cMsg As MSG
    Var Ret As Long

    Do While Api_PeekMessage(cMsg, 0, 0, 0, PM_REMOVE) <> 0
        Ret = Api_TranslateMessage(cMsg)
        Ret = Api_DispatchMessage(cMsg)
    Loop
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Counter As Long

    For Counter = 1 To 10000
        Select Case Index
            Case 0
                CallEvent
            Case 1
                OrgCallEvent
        End Select
        Text1.SetWindowText Str$(Counter)
    Next
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
```

```
Wend
Stop
End
```

CDドアの開閉(Ⅰ)

CDドアの開閉を実行します。API・F-BASICコマンドの両方をテストしています。
mciSendString MCIデバイスにコマンド文字列を送信
mciExecute MCIにアクセス




```
'=====================================
'= CDドアの開閉
'=====================================
#include "Windows.bi"

' 文字列を MCI に送信
Declare Function Api_mciSendString& Lib "winmm" Alias "mciSendStringA" (ByVal
lpstrCommand$, ByVal lpstrReturnString$, ByVal uReturnLength&, ByVal hwndCallback&)

' MCIにアクセス
Declare Function Api_mciExecute& Lib "winmm" Alias "mciExecute" (ByVal lpstrCommand$)

Var Shared Mci As Object
Mci.Attach GetDlgItem("MCI1")

'=====================================
'= APIでDoor Open
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    Ret = Api_mciSendString("set CDAudio door open", ByVal 0, 0, 0)
End Sub

'=====================================
'= APIでDoor Close
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Ret As Long

    Ret = Api_mciSendString("set CDAudio door closed", ByVal 0, 0, 0)
End Sub

'=====================================
'= F-BASICでDoor Open
'=====================================
Declare Sub Button3_on edecl ()
Sub Button3_on()
    Mci.SendString "Open CDAudio Alias CDDevice wait"
```

```
    Mci.SendString "Set CDDevice Door Open wait"
    Mci.SendString "Close CDDevice"
End Sub

'=====================================
'=  F-BASICでDoor Close
'=====================================
Declare Sub Button4_on edecl ()
Sub Button4_on()
    Mci.SendString "Open CDAudio Alias CDDevice wait"
    Mci.SendString "Set CDDevice Door Closed wait"
    Mci.SendString "Close CDDevice"
End Sub

'=====================================
'= APIでDoor Open
'=====================================
Declare Sub Button5_on edecl ()
Sub Button5_on()
    Var Ret As Long

    Ret = Api_mciExecute("set CDAudio door open")
End Sub

'=====================================
'= APIでDoor Close
'=====================================
Declare Sub Button6_on edecl ()
Sub Button6_on()
    Var Ret As Long

    Ret = Api_mciExecute("set CDAudio door closed")
End Sub

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

CDドアの開閉(II)

---

mciSendCommand 指定されたMCI(メディアコントロールインターフェイス)デバイスへ、コマンドメッセージを送信


```
'=====================================
'= CDドアの開閉(II)
'=    (mciSendCommand.bas)
'=====================================
#include "Windows.bi"

#define MCI_OPEN &H803                      'デバイスまたはデバイス要素を初期化する
#define MCI_OPEN_TYPE &H2000                'lpParamパラメータに指定する構造体の
                                            lpstrDeviceTypeメンバを有効にする
#define MCI_OPEN_SHAREABLE &H100            'デバイスまたはファイルを共有可能としてオープン
#define MCI_SET &H80D                       'デバイス情報を設定する
```

```
#define MCI_SET_DOOR_OPEN &H100              'メディアトレイをオープン
#define MCI_SET_DOOR_CLOSED &H200            'メディアトレイをクローズ
#define MCI_CLOSE &H804                      'デバイスまたはデバイス要素へのアクセスを解放する

Type MCI_OPEN_PARMS
    dwCallback       As Long
    wDeviceID        As Long
    lpstrDeviceType  As Long
    lpstrElementName As Long
    lpstrAlias       As Long
End Type

' 指定されたMCI(メディアコントロールインターフェイス)デバイスへ、コマンドメッセージを送信
Declare Function Api_mciSendCommand& Lib "winmm" Alias "mciSendCommandA" (ByVal
IDDevice&, ByVal uMsg&, ByVal fdwCommand&, ByRef dwParam As Any)

Var Shared Picture1 As Object
Var Shared Button1 As Object
Var Shared Button2 As Object
Var Shared Bitmap As Object

Picture1.Attach GetDlgItem("Picture1")
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14
BitmapObject Bitmap

Var Shared mop As MCI_OPEN_PARMS
Var Shared flg As Integer

'=====================================
'=
'=====================================
Declare Sub OpenClose ()
Sub OpenClose()
    If flg = 0 Then
        Bitmap.LoadFile "PC1.bmp"
    Else
        Bitmap.LoadFile "PC2.bmp"
    End If
    Picture1.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    flg = 0
    OpenClose
End Sub

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

    flg = 1
    OpenClose

    '構造体初期化
    mop.wDeviceID = 0
    mop.lpstrDeviceType = StrAdr("cdaudio" & Chr$(0))

    'ID取得
    Ret = Api_mciSendCommand(0, MCI_OPEN, MCI_OPEN_TYPE Or MCI_OPEN_SHAREABLE, mop)
```

```
    'CD-ROMドアオープン
    Ret = Api_mciSendCommand(mop.wDeviceID, MCI_SET, MCI_SET_DOOR_OPEN, ByVal 0)

    '解放
    Ret = Api_mciSendCommand(mop.wDeviceID, MCI_CLOSE, 0, ByVal 0)
End Sub

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

    flg = 0
    OpenClose

    '構造体初期化
    mop.wDeviceID = 0
    mop.lpstrDeviceType = StrAdr("cdaudio" & Chr$(0))

    'ID取得
    Ret = Api_mciSendCommand(0, MCI_OPEN, MCI_OPEN_TYPE Or MCI_OPEN_SHAREABLE, mop)

    'CD-ROMドアクローズ
    Ret = Api_mciSendCommand(mop.wDeviceID, MCI_SET, MCI_SET_DOOR_CLOSED, ByVal 0)

    '解放
    Ret = Api_mciSendCommand(mop.wDeviceID, MCI_CLOSE, 0, ByVal 0)
End Sub

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

Chr$(0)を取り除く

Chr$(0)を取り除きます。

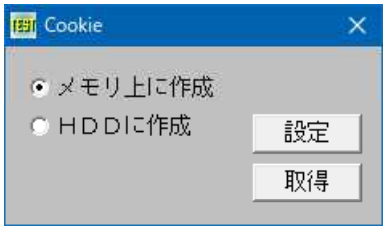
例ではabcdefgの文字列にChr$(0)を付加し、
abcdefgはTrimNull関数を通過した文字列です。
abcdefg¢は元の文字列ですがChr$(0)は表示できないので文字化けしています。
ファイルに書き込んでダンプで見ると0になっています。

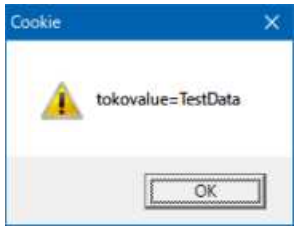

```
'=====================================
'= Chr$(0)を取り除く
'=====================================
Declare Function TrimNull (item As String) As String
Function TrimNull(item As String) As String
    Var ePos As Integer

    ePos = Instr(item, Chr$(0))
    If ePos Then
        TrimNull = Left$(item, ePos - 1)
```

```
    Else
        TrimNull = item
    End If
End Function

'------------------------------
Var Buff As String
Var strWK As String

buff = "abcdefg" & Chr$(0)
strWK = TrimNull(buff)

Print strWK
Print buff

Stop
End
```

Cookieの設定と取得

クッキーの設定と取得をテストします。
InternetSetCookie IEにCookie を設定
InternetGetCookie IEからCookie を取得





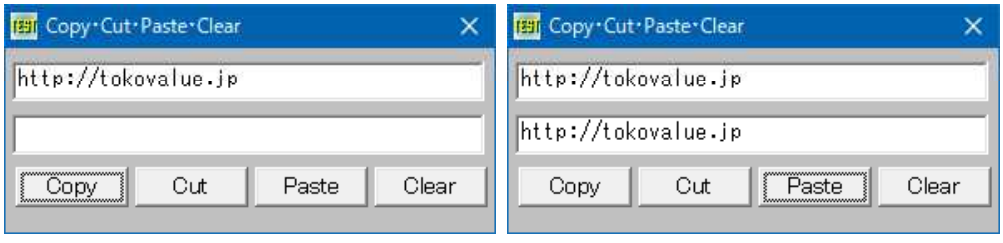
日付を設定した場合、下記のようにCookiesフォルダに保存されます。


```
'========================================
'= Cookieの設定と取得
'=    (InternetGetCookie.bas)
'========================================
#include "Windows.bi"

' IEにCookieを設定
Declare Function Api_InternetSetCookie& Lib "wininet" Alias "InternetSetCookieA" (ByVal
lpszUrlName$, ByVal lpszCookieName$, ByVal lpszCookieData$) As Long

' IEからCookieを取得
Declare Function Api_InternetGetCookie& Lib "wininet" Alias "InternetGetCookieA" (ByVal
lpszUrlName$, ByVal lpszCookieName$, ByVal lpszCookieData$, lpdwSize&) 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 Integer

    If Index = 0 Then

        'クッキーの作成(メモリ)
        Ret = Api_InternetSetCookie("http://www.testcom.index.htm", "tokovalue",
"TestData")
    Else

        '持続的なクッキーの作成(HDDに保存される)
        Ret = Api_InternetSetCookie("http://www.testcom.index.htm", "tokovalue",
"TestData; expires = Sat, 23-Sep-2006 06:00:00 GMT")
    End If

    If Ret = False Then
        A% = MessageBox(GetWindowtext, "失敗！", 0, 2)
    End If
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Buffer As String * 256
    Var Ret As Integer

    Ret = Api_InternetGetCookie("http://www.testcom.index.htm", "tokovalue", Buffer,
255)

    If Ret = False Then
        A% = MessageBox(GetWindowtext, "失敗！", 0, 2)
    Else
        A% = MessageBox(GetWindowtext, Buffer, 0, 2)
    End If
End Sub

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

Copy･Cut･Paste･Clear

F-BasicでのCopy･Cut･Paste･ClearをAPIで実行しています。
SendMessage ウィンドウにメッセージを送信
OpenClipboard クリップボードをオープン
EmptyClipboard クリップボードを空にする
CloseClipboard クリップボードをクローズ



```
'====================================
'= Copy･Cut･Paste･Clear
'=    (CopyPaste.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&)

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

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

#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_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_PASTE &H302                      'クリップボードからテキストをコピーした
#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

Var Shared Edit1 As Object
Var Shared Edit2 As Object
Var Shared Button(3) As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Edit2.Attach GetDlgItem("Edit2") : Edit2.SetFontSize 14
For i = 0 To 3
    Button(i).Attach GetDlgItem("Button" & Trim$(Str$(i + 1))) : Button(i).SetFontSize 14
Next

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

    'Edit1内のテキストを全選択
    Edit1.SetSelText 0, -1

    'Edit1の選択されたテキストをコピー(Edit1.Copyに相当)
    Ret = Api_SendMessage(Edit1.GethWnd, WM_COPY, 0, ByVal 0)
    Edit1.SetSelText -1, -1
End Sub

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

    'Edit1内のテキストを全選択
    Edit1.SetSelText 0, -1

    'Edit1の選択されたテキストを切り取り(Edit1.Cutに相当)
    Ret = Api_SendMessage(Edit1.GethWnd, WM_CUT, 0, ByVal 0)
    Edit1.SetSelText -1, -1
End Sub

'====================================
'= 貼り付け
'====================================
```

```
Declare Sub Button3_on edecl ()
Sub Button3_on()
    Var Ret As Long

    'Edit2をクリア
    Edit2.SetWindowtext ""

    'Edit2に貼り付け
    Ret = Api_SendMessage(Edit2.GethWnd, WM_PASTE, 0, ByVal 0)
End Sub

'=====================================
'= クリア
'=====================================
Declare Sub Button4_on edecl ()
Sub Button4_on()
    Var Ret As Long

    Edit2.SetSelText 0, -1

    'Edit2をクリア(Edit2.Clearに相当)
    Ret = Api_SendMessage(Edit2.GethWnd, WM_CLEAR, 0, ByVal 0)

    'クリップボード内のデータをクリア(CleaeCbに相当)
    Ret = Api_OpenClipBoard(GethWnd)
    Ret = Api_EmptyClipBoard()
    Ret = Api_CloseClipboard()
End Sub

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

[CTRL]+[矢印]キーで単語ごとにカーソル移動

InitCommonControlsEx 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
CreateWindowEx ウィンドウ(コントロール)を作成
SendMessage ウィンドウにメッセージを送信

コードで作成したコンボボックスのEditに、"http://tokovalue.jp/"の文字を入れています。
起動直後、[CTRL]+[矢印]キーでカーソルは文字の最後まで移動しますが、「実行」押下後は、スラッシュ、バックスラッシュ、ピリオドを区切り文字として[CTRL]+[矢印]キーで単語ごとにカーソル移動します。

カーソルは赤で表しています。

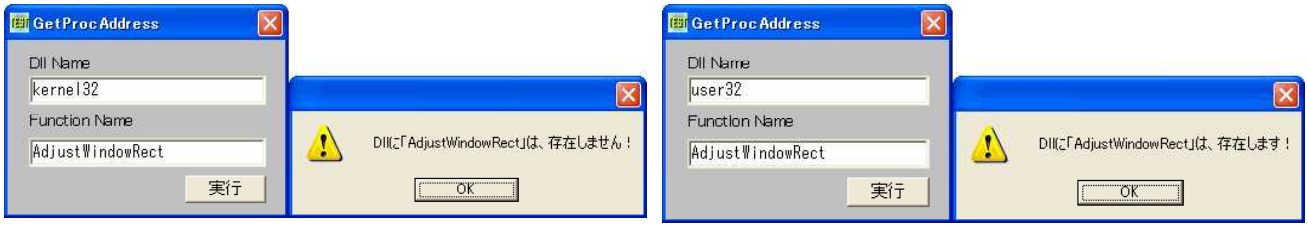

```
'=====================================
'= [CTRL]+[矢印]キーで単語ごとにカーソル移動
'=    (InitCommonControlsEx.bas)
'=====================================
#include "Windows.bi"

Type INITCOMMONCONTROLSEX
    dwSize      As Long
    dwICC       As Long
End Type
```

```
Type COMBOBOXEXITEM
    mask              As Long
    iItem             As Long
    pszText           As Long
    cchTextMax        As Long
    iImage            As Long
    iSelecteVarage    As Long
    iOverlay          As Long
    iIndent           As Long
    lParam            As Long
End Type

#define WC_COMBOBOXEX "ComboBoxEx32"        '拡張コンボボックス
#define ICC_USEREX_CLASSES &H200            '拡張コンボボックス
#define WS_VISIBLE &H10000000               '可視状態のウィンドウを作成する
#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 CBS_DROPDOWN &H2                    'CBS_SIMPLEで、リストはドロップダウンアイコンで表示する
#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 CBEM_SETEXTENDEDSTYLE &H40E         '(WM_USER + 14)
#define CBEM_INSERTITEM &H401               '(WM_USER + 1)
#define CB_SETCURSEL &H14E                  'コンボボックスのリストボックス内の文字列を選択する
#define CBES_EX_PATHWORDBREAKPROC &H4
#define CBEIF_TEXT &H1

' 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 Function Api_InitCommonControlsEx& Lib "comctl32" Alias "InitCommonControlsEx"
(lpInitCtrls As INITCOMMONCONTROLSEX)

' 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)

' 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)

Var Shared Button1 As Object

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

Var Shared hWindow As Long

'=======================================
'=
'=======================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var icc As INITCOMMONCONTROLSEX
    Var cbi As COMBOBOXEXITEM
    Var InitSuccess As Long
    Var WindowStyle As Long
    Var ItemIndex As Long
    Var ItemString As String
    Var Index As Long
    Var Ret As Long

    'コモンコントロールのクラスを指定
    icc.dwSize = Len(icc)
    icc.dwICC = ICC_USEREX_CLASSES

    'コモンコントロールのクラスを登録
    InitSuccess = Api_InitCommonControlsEx(icc)

    'コモンコントロールのクラスを登録できたとき
    If InitSuccess <> 0 Then

        'ウィンドウスタイルを設定
        WindowStyle = WS_CHILD Or WS_VISIBLE Or CBS_DROPDOWN
```

```
        'ウィンドウを作成
        hWindow = Api_CreateWindowEx(0, WC_COMBOBOXEX, "コンボボックスEx", WindowStyle, 10,
26, 216, 100, GethWnd, 0, GethInst, ByVal 0)
    End If

    'ComboboxExの末尾に項目追加する指定
    ItemIndex = (-1)

    '追加する文字列を指定
    ItemString = "http://tokovalue.jp/"

    '追加する項目情報を構造体に設定
    cbi.iItem = ItemIndex
    cbi.mask = CBEIF_TEXT
    cbi.pszText = StrAdr(ItemString)
    cbi.cchTextMax = Len(ItemString)

    'ComboboxExの初期値を設定
    Index = Api_SendMessage(hWindow, CBEM_INSERTITEM, 0, cbi)

    'ComboboxExの末尾に項目追加する指定
    ItemIndex = (-1)

    '追加する文字列を指定
    ItemString = "http://tokovalue.jp/"

    '追加する項目情報を構造体に設定
    cbi.iItem = ItemIndex
    cbi.mask = CBEIF_TEXT
    cbi.pszText = StrAdr(ItemString)
    cbi.cchTextMax = Len(ItemString)

    'ComboboxExインデックスを変更
    Ret = Api_SendMessage(hWindow, CB_SETCURSEL, Index, ByVal 0)
End Sub

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

    '単語ごとにカーソル移動する拡張スタイルを指定
    ExStyle = CBES_EX_PATHWORDBREAKPROC

    '拡張スタイルを設定
    Ret = Api_SendMessage(hWindow, CBEM_SETEXTENDEDSTYLE, 0, ByVal ExStyle)
End Sub

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

DLL内の関数アドレスを取得

GetModuleHandle 指定の実行モジュールのハンドルを取得
LoadLibrary DLLをロード
GetProcAddress 実行モジュール内の関数アドレスを取得
FreeLibrary ロードしたDLLの解放

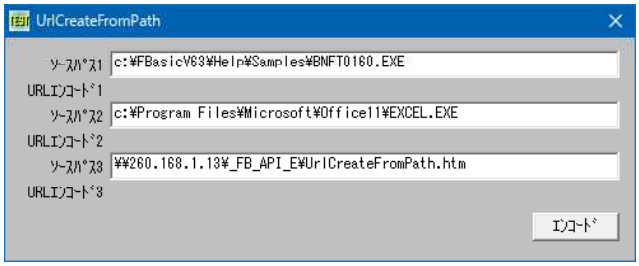


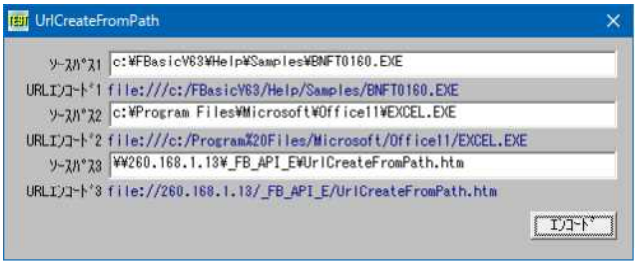

```
'=====================================
'= DLL内の関数アドレスを取得
'=    (GetProcAddress.bas)
'=====================================
#include "Windows.bi"

' 指定の実行モジュールのハンドルを取得
Declare Function Api_GetModuleHandle& Lib "Kernel32" Alias "GetModuleHandleA" (ByVal
ModuleName$)

' DLLをロード
Declare Function Api_LoadLibrary& Lib "Kernel32" Alias "LoadLibraryA" (ByVal
lpLibFileName$)

' 実行モジュール内の関数アドレスを取得
Declare Function Api_GetProcAddress& Lib "Kernel32" Alias "GetProcAddress" (ByVal
ModuleHandle&, ByVal ProcName$)

' ロードしたDLLの解放
Declare Sub Api_FreeLibrary Lib "Kernel32" Alias "FreeLibrary" (ByVal hLibModule&)

Var Shared Edit1 As Object
Var Shared Edit2 As Object
Var Shared Text1 As Object
Var Shared Text2 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
Text2.Attach GetDlgItem("Text2") : Text2.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Function IsFunctionExported(sFunction As String, sModule As String) As Integer
Function IsFunctionExported(sFunction As String, sModule As String) As Integer
    Var hMod As Long
    Var Loaded As Integer

    'DLLのハンドルを取得
    hMod = Api_GetModuleHandle(sModule)

    If hMod = 0 Then
        hMod = Api_LoadLibrary(sModule)
        If hMod Then Loaded = True
    End If
    If hMod Then
        If Api_GetProcAddress(hMod, sFunction) Then IsFunctionExported = True
    End If

    If Loaded Then Api_FreeLibrary(hMod)
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    If IsFunctionExported(Edit2.GetWindowText, Edit1.GetWindowText) Then
```

```
        A% = MessageBox("", "Dllに「" & Edit2.GetWindowText & "」は、存在します！", 0, 2)
    Else
        A% = MessageBox("", "Dllに「" & Edit2.GetWindowText & "」は、存在しません！", 0, 2)
    End If
End Sub

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

DOSパスから適切なURLパスに変換

UrlCreateFromPath DOSパスからURLパスに変換





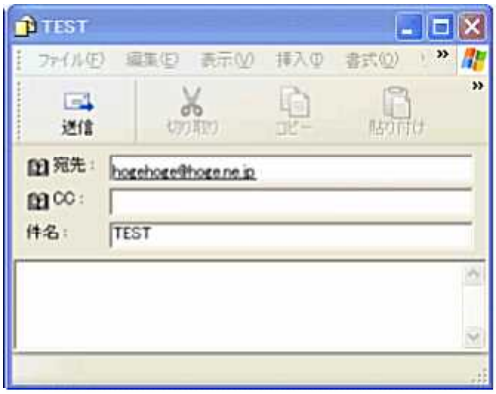

```
'=======================================
'= DOSパスから適切なURLパスに変換
'=    (UrlCreateFromPath.bas)
'=======================================
#include "Windows.bi"

#define ERROR_SUCCESS 0                    '正常終了の戻り値を示す
#define MAX_PATH 260                       '

' DOSパスから適切なURLパスに転換
Declare Function Api_UrlCreateFromPath& Lib "shlwapi" Alias "UrlCreateFromPathA" (ByVal
pszPath$, ByVal pszUrl$, pcchUrl&, ByVal dwReserved&)

Var Shared Text(8) As Object
Var Shared Edit(2) As Object
Var Shared Button1 As Object

For i = 0 To 8
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1))) : Text(i).SetFontSize 12
    If i < 3 Then
        Edit(i).Attach GetDlgItem("Edit" & Trim$(Str$(i + 1))) : Edit(i).SetFontSize 12
    End If
Next
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 12

'=======================================
'=
'=======================================
Declare Function CreateUrlFromPath(sPath As String) As String
Function CreateUrlFromPath(sPath As String) As String
    Var sUrl As String
    Var dwSize As Long

    If Len(sPath) > 0 Then
        sUrl = Space$(MAX_PATH)
        dwSize = Len(sUrl)
        If Api_UrlCreateFromPath(sPath, sUrl, dwSize, 0) = ERROR_SUCCESS Then
            CreateUrlFromPath = Left$(sUrl, dwSize)
```

```
        End If
    End If
End Function

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
   Edit(0).SetWindowText "c:¥FBasicV63¥Help¥Samples¥BNFT0160.EXE"
   Text(6).SetWindowText ""
   Edit(1).SetWindowText "c:¥Program Files¥Microsoft¥Office11¥EXCEL.EXE"
   Text(7).SetWindowText ""
   Edit(2).SetWindowText "¥¥260.168.1.13¥_FB_API_E¥UrlCreateFromPath.htm"
   Text(8).SetWindowText ""
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
   Var sPath As String
   Var sUrl As String

   sPath = Edit(0).GetWindowText
   sUrl = CreateUrlFromPath(sPath)
   Text(6).SetWindowText sUrl

   sPath = Edit(1).GetWindowText
   sUrl = CreateUrlFromPath(sPath)
   Text(7).SetWindowText sUrl

   sPath = Edit(2).GetWindowText
   sUrl = CreateUrlFromPath(sPath)
   Text(8).SetWindowText sUrl
End Sub

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

E-Mail(クリップボード利用のメッセージ作成)

Outlook Express の「メッセージの作成」本文入力部にクリップボードを利用してデータ(TEXT)を送ってみます。
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
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
ShellExecute 拡張子に関連付けられたプログラムを実行
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
keybd_event 特殊キーの状態を設定

1.本文を用意(範囲選択しクリップボードにコピー)
2.「メッセージの作成」ウィンドウを起動(ここでは「宛先」にFocusが当たっている)
3.本文入力部のハンドルを取得
4.上記ハンドルにマウスボタンクリック状態を送る
5.クリップボード内容を貼り付ける




```
'=======================================
'= E-Mail(クリップボード利用のメッセージ作成)
'=    (FindWindow2.bas)
'=======================================
#include "Windows.bi"
#include "File.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$,
ByVal lpWindowName$)

' 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 hWnd&,
ByVal hChildAfter&, ByVal lpszClass$, ByVal lpszWindow$)

' 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&)

' 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&)

' 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&)

' 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 HWND_TOPMOST (-1)                    'ウィンドウを常に最前面に配置
#define SWP_SHOWWINDOW &H40                  'ウィンドウを表示する
#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 VK_CONTROL &H11                      '[Ctrl]
#define VK_TAB &H9                           '[TAB]
#define VK_V &H56                            '(86)Vキー
#define KEYEVENTF_KEYUP &H2                  'キーを放す

#define WM_LBUTTONDOWN &H201                 '左のマウスボタンを押した
#define WM_LBUTTONUP &H202                   '左のマウスボタンが解放された

Var Shared Text1 As Object
Var Shared Text2 As Object
Var Shared Edit1 As Object
Var Shared Button(3) As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Text2.Attach GetDlgItem("Text2") : Text2.SetFontSize 14
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
```

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

Var Shared hParent As Long
Var Shared hIEServer As Long
Var Shared EMail As String

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

    EMail = ""
    EMail = EMail & "mailto:hogehoge@hoge.ne.jp"
    EMail = EMail & "?Subject=TEST"
    EMail = EMail & "&body="

    hParent = Api_FindWindow("ATH_Note", ByVal 0)

    If hParent = 0 Then
        Ret = Api_ShellExecute(GethWnd, "Open", EMail, ByVal 0, ByVal 0, SW_NORMAL)
        Wait 5
        hParent = Api_FindWindow("ATH_Note", ByVal 0)
    End If

    'ハンドル確認
    Text1.SetWindowText "hParent:" & "&&H" & Hex$(hParent)
    Text2.SetWindowText "hIEServer:" & "&&H" & Hex$(hIEServer)
End Sub

'=====================================
'= クリップボードへコピー
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    SetMapMode 0
    CLEARCB
    SETCBFORMAT "CF_TEXT"

    Edit1.SetSelText 0, -1
    Edit1.Copy
    Edit1.SetSelText -1, -1
End Sub

'=====================================
'=「Internet Explorer_Server」のハンドル取得
'= AH_Note
'=   └ ME_DocHost
'=       └ ##MimeEdit_Server
'=           └ Internet Explorer_Server
'=====================================
Declare Sub Button3_on edecl ()
Sub Button3_on()
    Var hChild As Long
    Var MyStr As String
    Var Ret As Long


    'メッセージの作成(Parent)
    hParent = Api_FindWindow("ATH_Note", ByVal 0)

    'Child1
    hChild = Api_FindWindowEx(hParent, 0, "ME_DocHost", ByVal 0)

    'Child2
    hChild = Api_FindWindowEx(hChild, 0, "##MimeEdit_Server", ByVal 0)
```

```
    'Child3(本文入力部)
    hIEServer = Api_FindWindowEx(hChild, 0, "Internet Explorer_Server", ByVal 0)

    'ハンドル確認
    Text1.SetWindowText "hParent:" & "&&H" & Hex$(hParent)
    Text2.SetWindowText "hIEServer:" & "&&H" & Hex$(hIEServer)
End Sub

'=====================================
'= 本文貼り付け
'=====================================
Declare Sub Button4_on edecl ()
Sub Button4_on()
    Var Ret As Long

    '「メッセージの作成」をアクティブに
    Ret = Api_SetWindowPos(hParent, HWND_TOPMOST, 0, 0, 0, 0, SWP_SHOWWINDOW Or SWP_NOMOVE
Or SWP_NOSIZE)

    '本文入力部にフォーカスセット
    Ret = Api_SendMessage(hIEServer, WM_LBUTTONDOWN, 0, 0)
    Ret = Api_SendMessage(hIEServer, WM_LBUTTONUP, 0, 0)

    'Clipboardデータを貼り付け(Ctrl+Vをシミュレート)
    Api_keybd_event VK_CONTROL, 0, 0, 0
    Api_keybd_event VK_V, 0, 0, 0
    Api_keybd_event VK_V, 0, KEYEVENTF_KEYUP, 0
    Api_keybd_event VK_CONTROL, 0, KEYEVENTF_KEYUP, 0
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 FileName As String
    Var CN As Long
    Var hFile As Byte
    Var Temp As String
    Var Buffer As String
    Var Ret As long

    CN = GetDropFileCount(DF)
    FileName = GetDropFileName(DF, 0)

    On Error GoTo *ErTrap

    hFile = FreeFile

    Open FileName For BinInp As hFile
        Buffer = Space$(Lof(hFile))
        fRead hFile, Buffer
        Buffer = jconv$(Buffer, 0, 2)
        Edit1.SetWindowText Buffer
    Close hFile
    Exit Sub

*ErTrap
    Close hFile
    Edit1.SetWindowText ""
    Resume *ErReturn
*ErReturn
    On Error GoTo 0
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
```

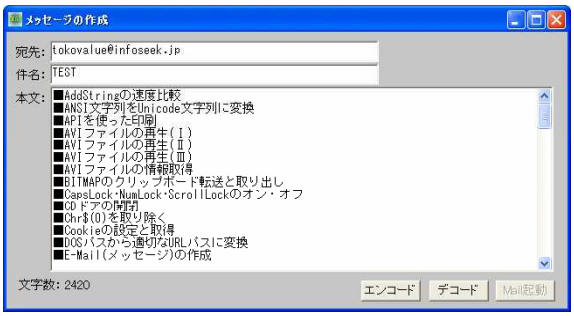
```
Wend
Stop
End
```

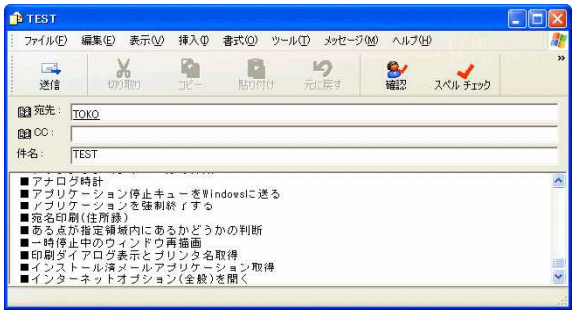
E-Mail(メッセージ)の作成

E-Mail(メッセージ)の作成の下準備をします。
ShellExecute 拡張子に関連付けられたプログラムを実行
UrlEscape %ASCIIに変換可能な安全ではない文字をエスケープシーケンスに変換
UrlUnescape エスケープシーケンスを普通の文字に変換

メーラーの「メッセージの作成」にデータを送ります。改行コードなど、そのまま送ることのできない文字をエスケープシーケンスに変換します。
「エンコード」クリックで変換された文字列の確認を、「デコード」クリックで元に戻ることの確認をしています。
文字数は、どの程度の文字数が送れるかをテストするためのものです。
フォームサイズの変更も可能です。
UrlEscapeは、1行998文字の制限がありました。1行998文字の処理を繰り返し、それらを加えていったらどうなるのかをテストしてみました。
区切る際、最後尾全角文字を分解してしまうようなことには対処していません。

「Mail起動」クリックでOutLookのメール作成を呼び出し、データをセットします。
※電子メールの仕様
1行の文字数は改行を含まないで78文字以下であるべきで、998文字を超えないこと。
改行は「復帰(CR)+改行(LF)」というシーケンスであること。






```
'========================================
'= E-Mail(メッセージ)の作成 II
'----------------------------------------
'= 1行998文字に分割
'= 入力(コピー＆ペーストを含む)文字数を監視しながらテストをしています。
'=    (SendMail2.bas)
'========================================
#include "Windows.bi"


' 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&)

#define OUT_OF_MEMORY_OR_RESOURCES 0        'メモリまたはリソースが足りない
#define ERROR_FILE_NOT_FOUND 2              'ファイルが見つからない
#define ERROR_PATH_NOT_FOUND 3              'パスが見つからない
#define ERROR_BAD_FORMAT 11                 '不正な形式の実行ファイル
```

```
#define SE_ERR_ACCESSDENIED 5                   'OSが指定したファイルへのアクセスを拒否
#define SE_ERR_ASSOCINCOMPLETE 27               'ファイルに関連づけられた物が不完全かまたは無効
#define SE_ERR_DDEBUSY 30                       '他のDDEプロセスが通信中だったので、DDE通信が完了できなかった
#define SE_ERR_DDEFAIL 29                       'DDE通信が失敗
#define SE_ERR_DDETIMEOUT 28                    'DDE通信でタイムアウトが発生
#define SE_ERR_DLLNOTFOUND 32                   '指定したDLLファイルが見つからなかった
#define SE_ERR_FNF 2                            'ファイルが見つからなかった
#define SE_ERR_NOASSOC 31                       '指定したファイルに関連づけられたアプリケーションが見つからなかった
#define SE_ERR_OOM 8                            '処理を完了するのに十分なメモリがなかった
#define SE_ERR_PNF 3                            '指定したパスが見つからなかった
#define SE_ERR_SHARE 26                         '共有違反が発生

#define SW_SHOWNORMAL 1                         'SW_RESTOREと同じ
#define c_CRLF "%0D%0A"
#define vbCrLf Chr$(13, 10)

' %ASCIIに変換可能な安全ではない文字をエスケープシーケンスに変換
Declare Function Api_UrlEscape& Lib "shlwapi" Alias "UrlEscapeA" (ByVal pszURL$, ByVal
pszEscaped$, pcchEscaped&, ByVal dwFlags&)

' エスケープシーケンスを普通の文字に変換
Declare Function Api_UrlUnescape& Lib "shlwapi" Alias "UrlUnescapeA" (ByVal pszURL$,
ByVal pszUnescaped$, pcchUnescaped&, ByVal dwFlags&)

#define MAX_PATH 5000                           '
#define ERROR_SUCCESS 0
#define URL_ESCAPE_SEGMENT_ONLY &H2000          'PSZのURLがそのセクションのみが含まれていることを示す
#define URL_ESCAPE_PERCENT &H1000               'URLのセグメントセクションにある任意の%文字を変換
#define URL_UNESCAPE_INPLACE &H100000           '
#define URL_INTERNAL_PATH &H800000
#define URL_DONT_ESCAPE_EXTRA_INFO &H2000000'クエリ内の文字の文字の変換を防止するために、
                                                URL_ESCAPE_SPACES_ONLYだけと連携して使用される
#define URL_ESCAPE_SPACES_ONLY &H4000000        'URLのクエリ部分でそれらのスペース文字を含む、そのエスケープシーケンスにのみ空白文字を変換します
#define URL_DONT_SIMPLIFY &H8000000

Var Shared Text(3) As Object
Var Shared Edit(2) As Object
Var Shared Button(2) As Object

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

Var Shared EMail As String

'=====================================
'=
'=====================================
Declare Sub Length_DSP edecl ()
Sub Length_DSP()
    Var Temp As String

    Temp = Edit(2).GetWindowText
    Text(3).SetWindowText "文字数:" & Str$(Len(Temp))
End Sub

'=====================================
'=
'=====================================
Declare Function Encode(sUrl As String) As String
Function Encode(sUrl As String) As String
```

```
    Var buff As String
    Var dwSize As Long
    Var dwFlags As Long

    If Len(sUrl) > 0 Then
        buff = Space$(MAX_PATH)
        dwSize = Len(buff)
        dwFlags = URL_DONT_SIMPLIFY

        If Api_UrlEscape(sUrl, buff, dwSize, dwFlags) = ERROR_SUCCESS Then
            Encode = Left$(buff, dwSize)
            Button(2).EnableWindow -1
        Else
            A% = MessageBox("", "制限文字数を越えています！", 0, 2)
        End If
    End If
End Function

'=====================================
'=
'=====================================
Declare Function Decode(sUrl As String) As String
Function Decode(sUrl As String) As String
    Var buff As String
    Var dwSize As Long
    Var dwFlags As Long

    If Len(sUrl) > 0 Then
        buff = Space$(MAX_PATH)
        dwSize = Len(buff)
        dwFlags = URL_DONT_SIMPLIFY

        If Api_UrlUnescape(sUrl, buff, dwSize, dwFlags) = ERROR_SUCCESS Then
            Decode = Left$(buff, dwSize)
            Button(2).EnableWindow 0
        Else
            A% = MessageBox("", "制限文字数を越えています！", 0, 2)
        End If
    End If
End Function

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Button(2).EnableWindow 0
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Temp As String
    Var Body As String
    Var i As Integer
    Var OneLine As Integer

    EMail = ""
    Body = ""
    OneLine = 998

    'E-Mail アドレス
    Temp = Trim$(Edit(0).GetWindowText)
    If Len(Temp) = 0 Then
        A% = MessageBox("Attention!", "E-Mail アドレスを入力してください！", 0, 2)
        Edit(0).SetFocus
        Exit Sub
    Else
```

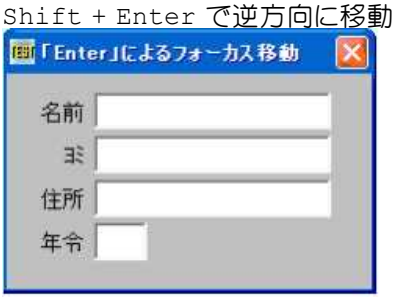
```
        EMail = EMail & "mailto:" & Temp
    End If

    '件名
    Temp = Trim$(Edit(1).GetWindowText)
    If Len(Temp) <> 0 Then
        EMail = EMail & "?Subject=" & Temp
    End If

    '本文
    Temp = rTrim$(Edit(2).GetWindowText)
    If Len(Temp) <> 0 Then
        For i = 1 To (Len(Temp) ¥ OneLine) + 1
            Body = Body & Encode(Mid$(Temp, (i - 1) * OneLine + 1, OneLine))
            Edit(2).SetWindowText Body
        Next

        EMail = EMail & "&body=" & Body
    End If

    Length_DSP
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Temp As String

    Temp = rTrim$(Edit(2).GetWindowText)

    If Len(Temp) <> 0 Then
        Temp = Decode(Temp)
        Edit(2).SetWindowText Temp
    End If
    Length_DSP
End Sub

'=====================================
'=
'=====================================
Declare Sub Button3_on edecl ()
Sub Button3_on()
    Var Ret As Long

    Ret = Api_ShellExecute(GethWnd, "Open", EMail, ByVal 0, ByVal 0, SW_SHOWNORMAL)

    If Ret <= 32 Then
        Select Case Ret
            Case OUT_OF_MEMORY_OR_RESOURCES
                A% = MessageBox("", "メモリまたはリソースが足りません。", 0, 2)
            Case ERROR_FILE_NOT_FOUND
                A% = MessageBox("", "ファイルが見つかりません。", 0, 2)
            Case ERROR_PATH_NOT_FOUND
                A% = MessageBox("", "パスが見つかりません。", 0, 2)
            Case ERROR_BAD_FORMAT
                A% = MessageBox("", "不正な形式の実行ファイルです。", 0, 2)
            Case SE_ERR_ACCESSDENIED
                A% = MessageBox("", "アクセスは拒否されました。", 0, 2)
            Case SE_ERR_ASSOCINCOMPLETE
                A% = MessageBox("", "関連付けが不完全です。", 0, 2)
            Case SE_ERR_NOASSOC
                A% = MessageBox("", "関連付けが指定されていません。", 0, 2)
            Case Else
                A% = MessageBox("", "ファイルを開けません。", 0, 2)
        End Select
    End If
End Sub
```

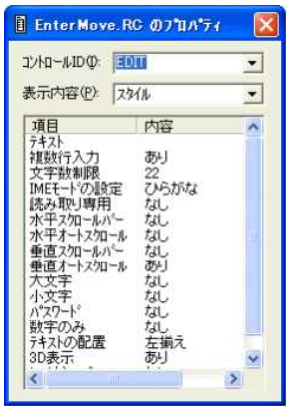
```
'=====================================
'=
'=====================================
Declare Sub Edit3_Change edecl ()
Sub Edit3_Change()
    Length_DSP
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Resize edecl ()
Sub MainForm_Resize()
    If GetWidth < 600 Or GetHeight < 350 Then
        SetWindowSize 600, 350
    End If

    Edit(2).SetWindowSize GetWidth - 74, GetHeight - 138
    Text(3).MoveWindow 14, GetHeight - 68
    Button(0).MoveWindow GetWidth - 244, GetHeight - 66
    Button(1).MoveWindow GetWidth - 166, GetHeight - 66
    Button(2).MoveWindow GetWidth - 90, GetHeight - 66
End Sub

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

Enterキーによる Edit間フォーカス移動（Ⅰ）

複数のエディットボックス間を「Enter」キーでフォーカス移動させます。
keybd_event 特殊キーの状態を設定
VK_TAB(&H9) [TAB]
KEYEVENTF_KEYUP(&H2) キーを放す

例では、「Enter」が押下されたとき、TabキーのKeyDownおよびKeyUpをシミュレートしています。
Shift + Enter で逆方向に移動します。





条件
EditBoxは、全て複数行入力 → あり、垂直オートスクロール → あり
EditBoxのコントロールIDは、4個とも「EDIT」とする。
文字数制限は、必要文字数+2（CrLfの2文字分）とする。

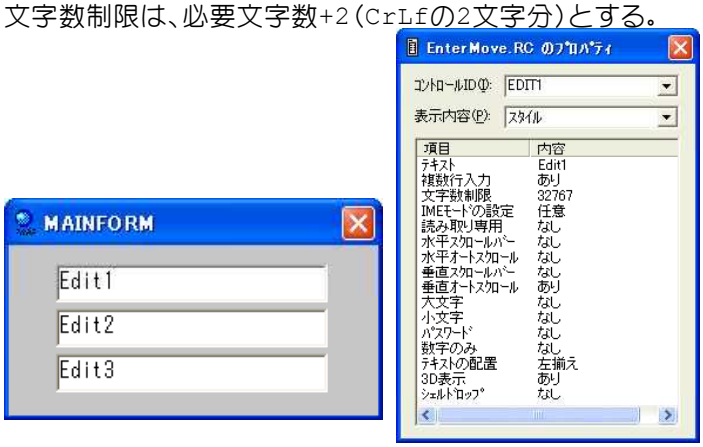
```
'=====================================
'= EnterKeyによるEditBox間移動
'= Edit1、Edit2、Edit3、Edit4はコントロールIDを全て「Edit」とする
'= 複数行入力（あり）、垂直オートスクロール（あり）
'= IMEモードを適宜設定。文字制限は必要最大文字数+2とする
'=====================================
```

```
#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 VK_TAB &H9                              '[TAB]
#define KEYEVENTF_KEYUP &H2                     'キーを放す

Var Shared Edit As Object

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

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Edit.SetFocus
End Sub

'=====================================
'= Enterを検知しTabを送る
'=====================================
Declare Sub EditControl_Change (hWnd As Long)
Sub EditControl_Change(hWnd As Long)
    Var Obj As Object
    Var Str As String
    Var EPos As Integer

    Obj.Attach hWnd
    Str = Obj.GetWindowText
    EPos = InStr(Str, Chr$(13))

    If EPos <> 0 Then
        Str = Mid$(Str, 1, EPos - 1) & Mid$(Str, EPos + 2)
        Obj.SetWindowText Str

        'TABをシミュレート
        Api_keybd_event VK_TAB, 0, 0, 0
        Wait 1
        Api_keybd_event VK_TAB, 0, KEYEVENTF_KEYUP, 0
    End If
End Sub

'=====================================
'= フォーカスのあるコントロールのハンドルを取得
'=====================================
Declare Sub Edit_Change edecl ()
Sub Edit_Change()
    EditControl_Change GetFocus
End Sub

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

Enterキーによる Edit間フォーカス移動(II)

複数のエディットボックス間を「Enter」キーでフォーカス移動させます。
Enter(CrLf)を検知し、次のEditBoxにフォーカスを当てています。
条件

EditBoxは、全て複数行入力 → あり
垂直オートスクロール → あり
文字数制限は、必要文字数+2(CrLfの2文字分)とする。




```
'=======================================
'= EnterKeyによるEditBox間移動
'= Edit1、Edit2、Edit3は
'= 複数行入力(あり)、垂直オートスクロール(あり)
'=======================================
#include "Windows.bi"

Var Shared Edit(2) As Object
For i% = 0 To 2
    Edit(i%).Attach GetDlgItem("Edit" & Trim$(Str$(i% + 1)))
    Edit(i%).SetFontSize 14
Next

#define CrLf Chr$(13, 10)

Var Shared Str(2) As String
Var Shared EPos As Integer

'=======================================
'=
'=======================================
Declare Sub MainForm_Start edecl ()
Sub mainForm_Start()
    Edit(0).SetFocus
End Sub

'=======================================
'=
'=======================================
Declare Sub Edit1_Change edecl ()
Sub Edit1_Change()
    Str(0) = Edit(0).GetWindowText
    EPos = InStr(Str(0), CrLf)
    If EPos <> 0 Then
        Str(0) = Mid$(Str(0), 1, EPos - 1) & Mid$(Str(0), EPos + 2)
        Edit(0).SetWindowText Str(0)
        Edit(1).SetFocus
    End If
End Sub

'=======================================
'=
'=======================================
Declare Sub Edit2_Change edecl ()
Sub Edit2_Change()
    Str(1) = Edit(1).GetWindowText
    EPos = InStr(Str(1), CrLf)
    If EPos <> 0 Then
        Str(1) = Mid$(Str(1), 1, EPos - 1) & Mid$(Str(1), EPos + 2)
        Edit(1).SetWindowText Str(1)
        Edit(2).SetFocus
```

```
    End If
End Sub

'=====================================
'=
'=====================================
Declare Sub Edit3_Change edecl ()
Sub Edit3_Change()
    Str(2) = Edit(2).GetWindowText
    EPos = InStr(Str(2), CrLf)
    If EPos <> 0 Then
        Str(2) = Mid$(Str(2), 1, EPos - 1) & Mid$(Str(2), EPos + 2)
        Edit(2).SetWindowText Str(2)
        Edit(0).SetFocus
    End If
End Sub

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

EXE･DLLからアイコンを取得する(Ⅰ)

EXE･DLLファイルからアイコンを取り出します。
ExtractIcon アイコンを取得
DrawIcon アイコンを描画
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

アイコンを取得するファイルパスをエディットボックスに入力し『取得』ボタンをクリックします。


```
'=====================================
'= EXE･DLLからアイコンを取り出す
'=    (ExtractIcon.bas)
'=====================================
#include "Windows.bi"

' 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&)

' アイコンを描画
Declare Function Api_DrawIcon& Lib "user32" Alias "DrawIcon" (ByVal hDC&, ByVal x&, ByVal
y&, ByVal exhIcon&)


' 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
```

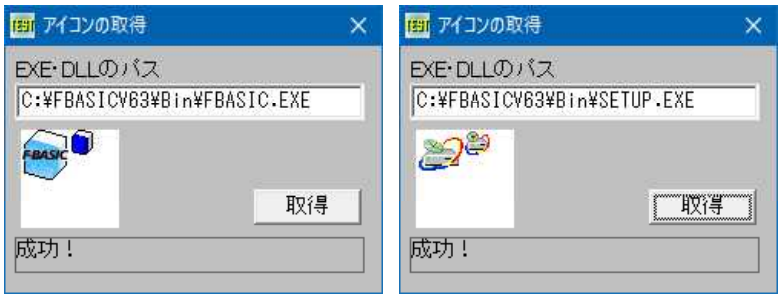
```
Var Shared Text1 As Object
Var Shared Text2 As Object
Var Shared Picture1 As Object
Var Shared Button1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Text2.Attach GetDlgItem("Text2") : Text2.SetFontSize 14
Picture1.Attach GetDlgItem("Picture1")
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Edit1.SetFocus
End Sub

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

    Picture1.cls
    pDC = Api_GetDC(Picture1.GethWnd)

    hIcon = Api_ExtractIcon(GethWnd, Edit1.GetWindowText, 0)

    If hIcon = 1 Then
        Text2.SetWindowText "EXE･DLLではありません"
        Exit Sub
    Else If hIcon = 0 Then
        Text2.SetWindowText "アイコンが存在しません"
        Exit Sub
    Else
        Ret = Api_DrawIcon(pDC, 0, 0, hIcon)
    End If

    If Ret <> 0 Then
        Text2.SetWindowText "成功！"
    Else
        Text2.SetWindowText "失敗！"
    End If

    Ret = Api_ReleaseDC(GethWnd, pDC)
End Sub

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

EXE･DLLからアイコンを取得する(II)

EXE･DLLファイルからアイコンを取り出します。
ExtractIconEx アイコンを取得
DrawIconEx アイコンを描画
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

アイコンを取得するファイルパスをエディットボックスにドラッグし『取得』ボタンをクリックします。
大きいアイコン、小さいアイコンを取得することができます。

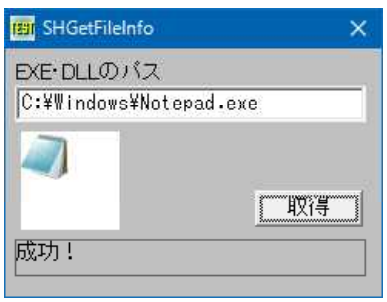



```
'=====================================
'= EXE･DLLからアイコンを取り出す(II)
'=    (ExtractIconEx.bas)
'=====================================
#include "Windows.bi"

' EXE･DLLのアイコンのハンドルを取得
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&)

' アイコンのハンドルを開放
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&)

#define DI_COMPAT 4                         'システムイメージで描画する
#define DI_DEFAULTSIZE 8                    'widthとheightが0である場合、アイコン(マウスカーソ
                                            ル)をデフォルトのサイズで描画
#define DI_IMAGE 2                          'イメージを使ってアイコン(またはマウスカーソル)を描画
#define DI_MASK 1                           'マスクを使ってアイコン(またはマウスカーソル)を描画
#define DI_NORMAL 3                         'DI_IMAGE と DI_MASK の組み合わせ

Var Shared Edit1 As Object
Var Shared Text1 As Object
Var Shared Text2 As Object
Var Shared Picture1 As Object
Var Shared Button1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Text2.Attach GetDlgItem("Text2") : Text2.SetFontSize 14
Picture1.Attach GetDlgItem("Picture1")
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var Shared FileName As String

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var pDC As Long                         'Picture1ハンドル
    Var lIcon As Long                       '32×32アイコン
    Var sIcon As Long                       '16×16アイコン
    Var Ret As Long
```

```
    Picture1.Cls
    pDC = Api_GetDC(Picture1.GethWnd)

    Ret = Api_ExtractIconEx(FileName, 0, lIcon, sIcon, 1)

    If Ret = 1 Then
        Text2.SetWindowText "EXE･DLLではありません"
        Exit Sub
    Else If Ret = 0 Then
        Text2.SetWindowText "アイコンが存在しません"
        Exit Sub
    Else
        Ret = Api_DrawIconEx(pDC, 0, 0, lIcon, 0, 0, ByVal 0, 0, DI_NORMAL)
        Ret = Api_DrawIconEx(pDC, 32, 0, sIcon, 0, 0, ByVal 0, 0, DI_NORMAL)
    End If

    If Ret <> 0 Then
        Text2.SetWindowText "成功！"
    Else
        Text2.SetWindowText "失敗！"
    End If

    Ret = Api_ReleaseDC(GetHwnd, pDC)
    Ret = Api_DestroyIcon(lIcon)
    Ret = Api_DestroyIcon(sIcon)
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
```

EXE･DLLからアイコンを取得する(III)

ファイルシステムオブジェクトの情報からアイコンを取得します。
SHGetFileInfo ファイルシステムオブジェクトの情報を取得
DrawIcon アイコンを描画する関数
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



```
'=====================================
'= EXE･DLLのアイコン取得
'=    (SHGetFileInfo.bas)
'=====================================
#include "Windows.bi"

#define MAX_PATH 260

' ファイルオブジェクトに関する情報を定義する構造体
Type SHFILEINFOA
    hIcon                   As Long
    iIcon                   As Long
    dwAttributes            As Long
    szDisplayName(MAX_PATH - 1) As byte
    szTypeName(80 - 1)      As byte
End Type

#define SHGFI_DISPLAYNAME &H200          'ディスプレイ名
#define SHGFI_EXETYPE &H2000             'EXEタイプ
#define SHGFI_ICON &H100                 'アイコン
#define SHGFI_LARGEICON &H0              '大きいアイコン
#define SHGFI_SHELLICONSIZE &H4          'シェルサイズアイコン
#define SHGFI_SMALLICON &H1              '小さいアイコン
#define SHGFI_SYSICONINDEX &H4000        'システムアイコンインデックス
#define SHGFI_TypeNAME &H400             'タイプ名

' ファイルシステムオブジェクトの情報を取得する関数の宣言
Declare Function Api_SHGetFileInfo& Lib "Shell32" Alias "SHGetFileInfoA" (ByVal
pszPath$, ByVal dwFileAttributes&, psfi As SHFILEINFOA, ByVal cbFileInfo&, ByVal uFlags&)

' アイコンを描画する関数の宣言
Declare Function Api_DrawIcon& Lib "user32" Alias "DrawIcon" (ByVal hDC&, ByVal x&, ByVal
y&, 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&)

Var Shared Picture1 As Object
Var Shared Edit1 As Object
Var Shared Text2 As Object
Var Shared Button1 As Object

Picture1.Attach GetDlgItem("Picture1")
Edit1.Attach GetDlgItem("Edit1") : Edit1.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 phDC As Long
    Var TargetFileName As String
    Var TargetInfo As Long
    Var ShellFileInfo As SHFILEINFOA
    Var Ret As Long
    Picture1.cls
    phDC = Api_GetDC(Picture1.GethWnd)            'Picture1のDC取得
    TargetFileName = GetDlgItemText("Edit1")      '対象のファイル名を指定
    Edit1.SetWindowText TargetFileName            'EditBoxにパス表示
    TargetInfo = SHGFI_ICON Or SHGFI_LARGEICON    '大きいアイコンの取得を指定
                                                  'ファイルに関する情報を取得
    Ret = Api_SHGetFileInfo(TargetFileName, 0, ShellFileInfo, Len(ShellFileInfo),
TargetInfo)
    Ret = Api_DrawIcon(phDC, 0, 0, ShellFileInfo.hIcon) '大きいアイコンを描画
```

```
    If Ret <> 0 Then
        Text2.SetWindowText "成功！"
    Else
        Text2.SetWindowText "失敗！"
    End If

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

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

EXE･DLLからアイコンを取得する(IV)

EXE･DLLからアイコンを取得します。
ExtractIcon ファイルに含まれるアイコンを描画
DrawIcon アイコンを描画
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

システムフォルダにあるc:¥windows¥system32¥shell32.dllに含まれているアイコンを取得しています。
システムフォルダは、明示的に設定しています。(Window2000ではc:¥winnt¥)

起動直後 → 指定ファイルには238個のアイコンが含まれており60番目のアイコンを表示しています。▲▼で順次呼び出し表示します。

```
'=======================================
'= EXE･DLLからアイコンを取得する(IV)
'=    (ExtractIcon2.bas)
'=======================================
#include "Windows.bi"

' アイコンを描画
Declare Function Api_DrawIcon& Lib "user32" Alias "DrawIcon" (ByVal hDC&, ByVal x&, ByVal
y&, ByVal hIcon&)

' ファイルに含まれるアイコンを描画
Declare Function Api_ExtractIcon& Lib "Shell32" Alias "ExtractIconA" (ByVal hInst&,
ByVal lpszExeFileName$, ByVal nIconIndex&)

' 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&)

Var Shared Picture1 As Object
Var Shared Text(4) As Object
```

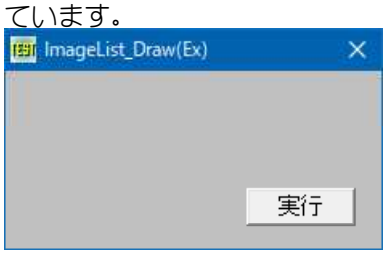
```
Var Shared Button(2) As Object
Var Shared sIconFile As String
Var Shared hDC As Long
Var Shared picH As Long
Var Shared picW As Long
Var Shared Value As Integer
Var Shared Min As Integer
Var Shared Max As Integer

'====================================
'=
'====================================
Declare Sub Mainform_Start edecl ()
Sub Mainform_Start()
    Picture1.Attach GetDlgItem("Picture1")
    For i = 0 To 4
        Text(i).Attach GetDLgItem("Text" & Trim$(Str$(i+1)))
        Text(i).SetFontSize 14
    Next
    For i = 0 To 2
        Button(i).Attach GetDlgItem("Button" & Trim$(Str$(i+1)))
        Button(i).SetFontSize 14
    Next

    hDC = Api_GetDC(Picture1.GethWnd)
    picW = Picture1.GetWidth
    picH = Picture1.GetHeight

    Button(1).EnableWindow 0
    Button(2).EnableWindow 0
End Sub

'====================================
'=
'====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hIcon As Long
    Var nIconCount As Long

    sIconFile = GetDlgItemText("Text1")

    nIconCount = Api_ExtractIcon(0, sIconFile, -1)

    If nIconCount > 0 Then

        Min = 0
        Max = nIconCount - 1
        Value = 0

        Text(0).SetWindowText sIconFile
        Text(1).SetWindowText Trim$(Str$(nIconCount)) & " Icons"
    End If

    Button(1).EnableWindow -1
    Button(2).EnableWindow -1
End Sub

'====================================
'= ▲
'====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var hIcon As Long
    Var x As Long
    Var y As Long
    Var Ret As Long
    x = (picH - 32) / 2
    y = (picW - 32) / 2
```

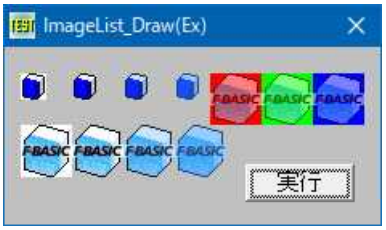
```
    If Value <= Max Then
        Picture1.cls
        hIcon = Api_ExtractIcon(0, sIconFile, Value)

        Ret = Api_DrawIcon(hDC, x, y, hIcon)
        Ret = Api_DestroyIcon(hIcon)

        Text(2).SetWindowText "Icon # " & Trim$(Str$(Value))
        Value = value + 1 : If Value > Max Then Value = Max
    End If
End Sub

'=====================================
'= ▼
'=====================================
Declare Sub Button3_on edecl ()
Sub Button3_on()
    Var hIcon As Long
    Var x As Long
    Var y As Long
    Var Ret As Long

    x = (picH - 32) / 2
    y = (picW - 32) / 2

    If Value >= Min Then

        Picture1.cls
        hIcon = Api_ExtractIcon(0, sIconFile, Value)

        Ret = Api_DrawIcon(hDC, x, y, hIcon)
        Ret = Api_DestroyIcon(hIcon)

        Text(2).SetWindowText "Icon # " & Trim$(Str$(Value))
        Value = value - 1 : If Value < 0 Then Value = 0
    End If
End Sub

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

EXE･DLLからアイコン取得（Ｖ）

EXE･DLLからアイコンを取得し、ImageList_Drawで描画します。
SHGetFileInfo ファイルシステムオブジェクトの情報を取得
ImageList_Draw イメージを描画
ImageList_DrawEx デバイスコンテキストにイメージリストのアイテムを描画（オフセット･背景色･前景色を指定できる）
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

EXEから小さいアイコン、および大きいアイコンを取得し、それぞれをそのまま･透過･25%ブレンド･50%ブレンド描画しています。





```
'=====================================
'= EXE･DLLからアイコン取得（Ｖ）
'=    (ImageList_Draw.bas)
'=====================================
#include "Windows.bi"

#define SHGFI_DISPLAYNAME &H200             'ディスプレイ名
#define SHGFI_EXEType &H2000                'EXEタイプ
#define SHGFI_LARGEICON &H0                 '大きいアイコン
#define SHGFI_SHELLICONSIZE &H4             'シェルサイズアイコン
#define SHGFI_SMALLICON &H1                 '小さいアイコン
#define SHGFI_SYSICONINDEX &H4000           'システムアイコンインデックス
#define SHGFI_TYPENAME &H400                'タイプ名
#define ILD_TRANSPARENT &H1                 'マスクを使って表示（背景色を無視）
#define ILD_BLEND50 &H4                     'システムハイライト色を50%ブレンド
#define ILD_BLEND25 &H2                     'システムハイライト色を25%ブレンド
#define CLR_NONE &HFFFFFFFF
#define CLR_DEFAULT &HFF000000
#define MAX_PATH 260

Type SHFILEINFO
    hIcon           As Long                 'アイコン
    iIcon           As Long                 'アイコンインデックス
    dwAttributes    As Long                 '属性
    szDisplayName   As String * MAX_PATH    'デスプレイ名（パス名）
    szTypeName      As Long                 'タイプ名
End Type

' ファイルシステムオブジェクトの情報を取得
Declare Function Api_SHGetFileInfo& Lib "shell32" Alias "SHGetFileInfoA" (ByVal
pszPath$, ByVal dwFileAttributes&, psfi As SHFILEINFO, ByVal cbFileInfo&, ByVal uFlags&)

' イメージを描画
Declare Function Api_ImageList_Draw& Lib "comctl32" Alias "ImageList_Draw" (ByVal himl&,
ByVal i&, ByVal hdcDst&, ByVal x&, ByVal y&, ByVal fStyle&)

' デバイスコンテキストにイメージリストのアイテムを描画（オフセット･背景色･前景色を指定できる）
Declare Function Api_ImageList_DrawEx& Lib "comctl32" Alias "ImageList_DrawEx" (ByVal
himl&, ByVal index&, ByVal hdcDst&, ByVal x&, ByVal y&, ByVal dx&, ByVal dy&, ByVal rgbBk&,
ByVal rgbFg&, ByVal fStyle&)

' 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 Button1 As Object

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

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var ePath As String
    Var hDC As Long
    Var hImage As Long
    Var SFI As SHFILEINFO
    Var Ret As Long

    ePath = "C:¥FBASICV63¥BIN¥FBASIC.exe"
    hDC = Api_GetDC(GethWnd)

    '小さいアイコンを含んでいるシステムイメージリストのハンドルを取得
    hImage = Api_SHGetFileInfo(ePath, ByVal 0, SFI, Len(SFI), SHGFI_SYSICONINDEX Or
SHGFI_SMALLICON)
```

```
    '小さいアイコン描画
    Ret = Api_ImageList_Draw(hImage, SFI.iIcon, hDC, 10, 16, 0)
    Ret = Api_ImageList_Draw(hImage, SFI.iIcon, hDC, 42, 16, ILD_TRANSPARENT)
    Ret = Api_ImageList_Draw(hImage, SFI.iIcon, hDC, 74, 16, ILD_TRANSPARENT Or
ILD_BLEND25)
    Ret = Api_ImageList_Draw(hImage, SFI.iIcon, hDC, 106, 16, ILD_TRANSPARENT Or
ILD_BLEND50)

    '大きいアイコンを含んでいるシステムイメージリストのハンドルを取得
    hImage = Api_SHGetFileInfo(ePath, ByVal 0, SFI, Len(SFI), SHGFI_SYSICONINDEX Or
SHGFI_LARGEICON)

    '大きいアイコン描画
    Ret = Api_ImageList_Draw(hImage, SFI.iIcon, hDC, 10, 48, 0)
    Ret = Api_ImageList_Draw(hImage, SFI.iIcon, hDC, 42, 48, ILD_TRANSPARENT)
    Ret = Api_ImageList_Draw(hImage, SFI.iIcon, hDC, 74, 48, ILD_TRANSPARENT Or
ILD_BLEND25)
    Ret = Api_ImageList_Draw(hImage, SFI.iIcon, hDC, 106, 48, ILD_TRANSPARENT Or
ILD_BLEND50)

    '背景色を指定
    Ret = Api_ImageList_DrawEx(hImage, SFI.iIcon, hDC, 128, 16, 32, 32, &HFF, 0,
ILD_BLEND50)
    Ret = Api_ImageList_DrawEx(hImage, SFI.iIcon, hDC, 160, 16, 32, 32, &HFF00, 0,
ILD_BLEND50)
    Ret = Api_ImageList_DrawEx(hImage, SFI.iIcon, hDC, 192, 16, 32, 32, &HFF0000, 0,
ILD_BLEND50)
    Ret = Api_ReleaseDC(GwthWnd, hDC)
End Sub

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

EXE･DLLからアイコン取得(VI)

EXE･DLLからアイコンを取得します。
ExtractAssociatedIcon 関連付けられている実行可能ファイルに含まれるアイコンハンドルを取得
DrawIconEx アイコンを描画
DestroyIcon アイコンのハンドルを解放

例では、c:¥fbasicv63¥bin¥fbasic.exeからアイコンを取得表示しています。色々な方法があるものですネ！


```
'=====================================
'= EXE･DLLからアイコン取得(VI)
'=    (ExtractAssociatedIcon.bas)
'=====================================
#include "Windows.bi"

#define DI_COMPAT 4                         'システムイメージで描画する
#define DI_DEFAULTSIZE 8                    'width パラメータとheightパラメータに0が指定されて
                                            いる場合、アイコンまたはマウスカーソルをデフォルト･
#define DI_IMAGE 2                          'イメージを使ってアイコン(またはマウスカーソル)を描画
#define DI_MASK 1                           'マスクを使ってアイコン(またはマウスカーソル)を描画
```

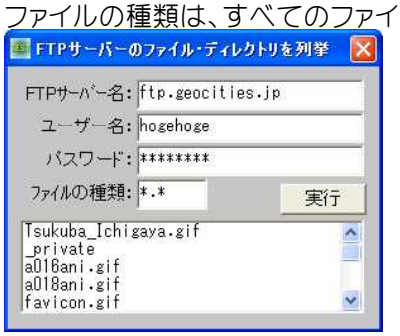
```
#define DI_NORMAL 3                         'DI_IMAGE と DI_MASK の組み合わ

' 関連付けられている実行可能ファイルに含まれるアイコンハンドルを取得
Declare Function Api_ExtractAssociatedIcon& Lib "shell32" Alias
"ExtractAssociatedIconA" (ByVal hInst&, ByVal lpIconPath$, lpiIcon&)

' 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&)

' 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&)

Var Shared Picture1 As Object
Var Shared Button1 As Object

Picture1.Attach GetDlgItem("Picture1")
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 hpDC As Long
    Var hIcon As Long
    Var Ret As Long

    hpDC = Api_GetDC(Picture1.GethWnd)
    Picture1.cls
    hIcon = Api_ExtractAssociatedIcon(GethInst, "c:¥fbasicv63¥bin¥fbasic.exe", 0)
    Ret = Api_DrawIconEx(hpDC, 0, 0, hIcon, 32, 32, 0, 0, DI_NORMAL)
    Ret = Api_DestroyIcon(hIcon)
    Ret = Api_ReleaseDC(Picture1.GethWnd, hpDC)
End Sub

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

FTPサーバーのディレクトリ･ファイルを列挙

FTPサーバーにあるディレクトリ、ファイルを列挙します。
InternetOpen インターネットのハンドルを作成
InternetConnect インターネットサーバーに接続し、ハンドルを返す
InternetCloseHandle インターネット関数のハンドルをクローズ

FtpFindFirstFile 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
InternetFindNextFile 検出したファイルの次を検出

例では、geocitiesのFTPサーバーにあるディレクトリおよびファイルを列挙しています。当然ユーザー名とパスワードは本物を入力します。
ファイルの種類は、すべてのファイルおよびwavファイルの列挙例です。



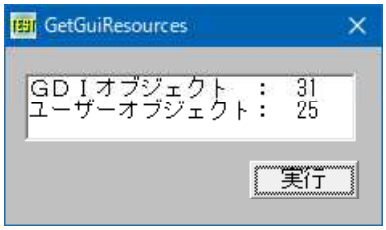

```
'=====================================
'= FTPサーバーのディレクトリ・ファイルを列挙
'=    (FtpFindFirstFile.bas)
'=====================================
#include "Windows.bi"

' インターネットのハンドルの作成
Declare Function Api_InternetOpen& Lib "wininet" Alias "InternetOpenA" (ByVal sAgent$,
ByVal lAccessType&, ByVal sProxyName$, ByVal sProxyBypass$, ByVal lFlags&)

' インターネット上のサーバに接続し、ハンドルを返す
Declare Function Api_InternetConnect& Lib "wininet" Alias "InternetConnectA" (ByVal
hSession&, ByVal sServerName$, ByVal nServerPort&, ByVal sUsername$, ByVal sPassword$,
ByVal lService&, ByVal lFlags&, ByVal lContext&)

' Win32インターネット関数のハンドルのクローズ
Declare Function Api_InternetCloseHandle% Lib "wininet" Alias "InternetCloseHandle"
(ByVal hInet&)

' 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_FtpFindFirstFile& Lib "wininet" Alias "FtpFindFirstFileA" (ByVal
hConnect&, ByVal lpszSearchFile$, lpFindFileData As Any, ByVal dwFlags&, ByVal
dwContext&)

' 検出したファイルの次を検出
Declare Function Api_InternetFindNextFile& Lib "wininet" Alias "InternetFindNextFileA"
(ByVal hConnect&, lpFindFileData As Any)

Type FILETIME
    dwLowDateTime     As Long
    dwHighDateTime    As Long
End Type
#define MAX_PATH 260

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
    cAlternateFileName      As String * 16
End Type

#define INTERNET_OPEN_TYPE_DIRECT 1         '直接接続
#define INTERNET_INVALID_PORT_NUMBER 0      '
#define INTERNET_SERVICE_FTP 1              'FTPを指定
#define INTERNET_FLAG_RELOAD -2147483648    'ローカルのキャッシュを無視し、常にサーバからデータ取得
```

```
Var Shared Edit(3) As Object
Var Shared Text(3) As Object
Var Shared List1 As Object
Var Shared Button1 As Object

For i = 0 To 3
    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
List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var wfd As WIN32_FIND_DATA
    Var hInet As Long
    Var hConnect As Long
    Var hFile As Long
    Var Ret As Long

    Button1.EnableWindow 0
    List1.Resetcontent

    '初期化
    hInet = Api_InternetOpen("", INTERNET_OPEN_TYPE_DIRECT, ByVal 0, ByVal 0, 0)

    SetMousePointer 2
    If hInet <> 0 Then

        'FTP接続処理
        hConnect = Api_InternetConnect(hInet, Edit(0).GetWindowText,
INTERNET_INVALID_PORT_NUMBER, Edit(1).GetWindowText, Edit(2).GetWindowText,
INTERNET_SERVICE_FTP, 0, 0)

        If hConnect <> 0 Then

            'FTPサーバーからファイルを検索
            hFile = Api_FtpFindFirstFile(hConnect, Edit(3).GetWindowText, wfd,
INTERNET_FLAG_RELOAD, 0)

            Do
                '検索結果をリストボックスへ追加
                List1.AddString Left$(wfd.cFileName, InStr(wfd.cFileName, Chr$(0)) - 1)

                'ファイルの検索を継続
                Ret = Api_InternetFindNextFile(hFile, wfd)
            Loop Until Ret = 0
        End If
    End If

    SetMousePointer 0

    'ハンドルを閉じる
    If hInet <> 0 Then
        Ret = Api_InternetCloseHandle(hInet)
    End If

    If hConnect <> 0 Then
        Ret = Api_InternetCloseHandle(hConnect)
    End If

    If hConnect <> 0 Then
        Ret = Api_InternetCloseHandle(hFile)
    End If
```

```
    Button1.EnableWindow -1
End Sub

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

GDIオブジェクトのハンドル数取得

指定したプロセスのGDIオブジェクトおよびUSERオブジェクトの数を取得します。
GetGuiResources 指定されたプロセスが使っているGUIオブジェクトのハンドルを返す
GetCurrentProcess 現在のプロセスに対応する疑似ハンドルを取得




```
'=====================================
'= GDIオブジェクトのハンドル数取得
'= Windows NT: バージョン 5.0 以降
'=    (GetGuiResources.bas)
'=====================================
#include "Windows.bi"

#define GR_GDIOBJECTS 0                    'GDIオブジェクトの数を返す
#define GR_USEROBJECTS 1                   'USERオブジェクトの数を返す

' 指定されたプロセスが使っているGUIオブジェクトのハンドルを返す
Declare Function Api_GetGuiResources& Lib "user32" Alias "GetGuiResources" (ByVal
hProcess&, ByVal uiFlags&)

' 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" ()

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 txt As String

    txt = txt & "GDIオブジェクト  :" & Format$(Api_GetGuiResources(Api_GetCurrentProcess,
GR_GDIOBJECTS), "####") & Chr$(13, 10)
    txt = txt & "ユーザーオブジェクト:" & Format$(Api_GetGuiResources(Api_GetCurrentProcess,
GR_USEROBJECTS), "####")
    Text1.SetWindowText txt
End Sub

'=====================================
'=
'=====================================
While 1
```

```
    WaitEvent
Wend
Stop
End
```

GUID(ユニークな128ビット値)を生成

GUID(Global Unique Identifier)2つ以上のアイテムが同じ値を持つことがない一意な識別子(128ビット)を生成します。
CoCreateGuid GUID(ユニークな128ビット値)を生成

生成する度に値は変わります。

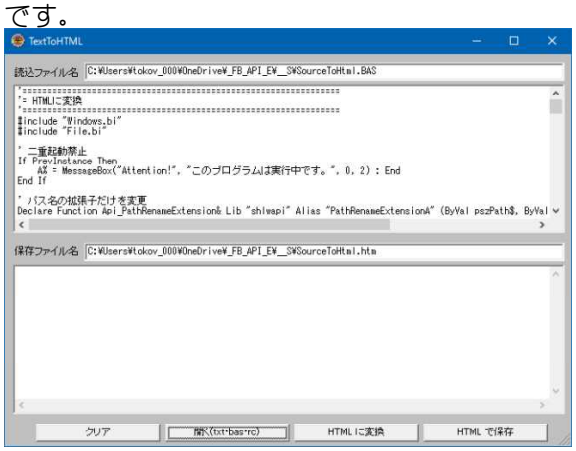

```
'=====================================
'= GUID(ユニークな128ビット値)を生成
'=    (CoCreateGuid.bas)
'=====================================
#include "Windows.bi"

' GUID(ユニークな128ビット値)を生成
Declare Function Api_CoCreateGuid& Lib "ole32" Alias "CoCreateGuid" (pGuid 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 id(15) As Byte
    Var i As Long
    Var GUID As String

    '正常な場合は0を返す
    If Api_CoCreateGuid(id(0)) = 0 Then
        For i = 0 To 15
            If id(i) < 16 Then
                GUID = GUID & "0" & Hex$(id(i))       '&HF以下の場合[0]を付加
            Else
                GUID = GUID & Hex$(id(i))
            End If
        Next i
    End If

    GUID = Left$(GUID, 8) & "-" & Mid$(GUID, 9, 4) & "-" & Mid$(GUID, 13, 4) & "-" &
Mid$(GUID, 17, 4) & "-" & Right$(GUID, 12)

    Text2.SetWindowText GUID
End Sub

'=====================================
'=
'=====================================
While 1
```

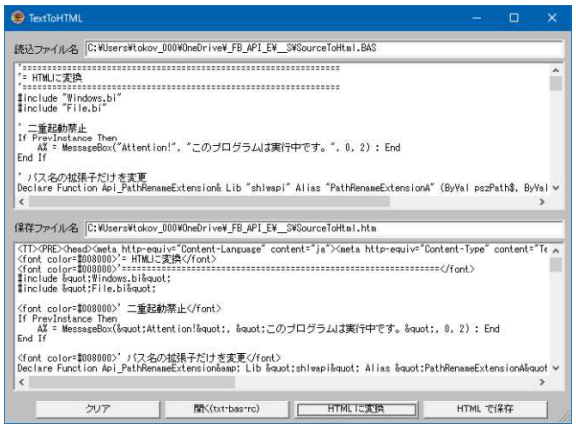
```
    WaitEvent
Wend
Stop
End
```

HTMLに変換

読み込んだファイルをHTMLに変換し保存します。
PathRenameExtension パス文字列から拡張子のみ変更する関数
PathRemoveExtension パス文字列から拡張子を取り除く関数
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

ホームページにTipsを載せる際、注釈部の色を変えた方が見やすくなるかナと思い作成してみました。Text3はサイズグリップ用です。
Edit1、Edit2に水平スクロールを「あり」に変更しました。
「HTMLに変換」のコード(bas)を読み込んでHTMLに変換した状態。注釈部「'」を見つけるとその行を暗緑色で表示させます。<font color=#008000>部
変換後のファイルは拡張子を「.htm」に自動変換し保存します。下段のコードは作成されたファイルを読み込んだものです。






```
'========================================
'= HTMLに変換
'=    (SourceToHtml.bas)
'========================================
#include "Windows.bi"
#include "File.bi"

' 二重起動禁止
If PrevInstance Then
    A% = MessageBox("Attention!", "このプログラムは実行中です。", 0, 2) : End
End If

' パス名の拡張子だけを変更
Declare Function Api_PathRenameExtension& Lib "shlwapi" Alias "PathRenameExtensionA"
(ByVal pszPath$, ByVal pszExt$)

' 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                           '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

#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 Edit(3) As Object
Var Shared Text(2) As Object
Var Shared Button(3) As Object

For i = 0 To 3
    Edit(i).Attach GetDlgItem("Edit" & Trim$(Str$(i + 1)))
    Edit(i).SetFontSize 12
Next
For i = 0 To 2
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1)))
    If i < 2 Then
        Text(i).SetFontSize 12
    End if
Next
For i = 0 To 3
    Button(i).Attach GetDlgItem("Button" & Trim$(Str$(i + 1)))
    Button(i).SetFontSize 12
Next

Var Shared FileName As String               'ファイル名
Var Shared zX As Integer                    'サイズグリップX軸
Var Shared zY As Integer                    'サイズグリップY軸

'=====================================
'= サイズグリップ位置計算
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var fs As Integer

    fs = 18
    Text(2).SetFontSize fs
    Text(2).SetWindowSize fs, fs
    Text(2).SetFontName "Marlett"
    Text(2).SetWindowText "o"

    ShowWindow -1

    zX = Api_GetSystemMetrics(SM_CXFRAME) * 2 + 1
    zY = Api_GetSystemMetrics(SM_CYFRAME) * 2 + Api_GetSystemMetrics(SM_CYBORDER) +
Api_GetSystemMetrics(SM_CYSIZE) + 1
End Sub

'=====================================
'= サイズグリップ処理
'=====================================
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(Text(2).GethWnd, WM_NCLBUTTONDOWN, HTBOTTOMRIGHT, 0)
    End if
End Sub

'=====================================
'= 注釈を暗緑色で
'=====================================
Declare Function Comments(txt As String) As String
Function Comments(txt As String) As String
    Var ePos As Integer
    Var ch As String
    Var quote As Integer
```

```
    For ePos = 1 To Len(txt)
        ch = Mid$(txt, ePos, 1)
        If ch = """" Then
            quote = Not quote
        Else If (ch = "'") And (Not quote) Then
            Exit For
        End if
    Next ePos

    If ePos <= Len(txt) Then

        '暗緑色で表示
        Comments = Left$(txt, ePos - 1) & "<font color=" & "#008000" & ">'" & Mid$(txt, ePos
+ 1) & "</font>"

        '太字で表示する場合
        'Comments = Left$(txt, ePos - 1) & "<B>'" & Mid$(txt, ePos + 1) & "</B>"
    Else
        Comments = txt
    End if
End Function

'=======================================
'= 置換関数
'=======================================
Declare Function ReplaceString(txt As String, from_str As String, to_str As String) As
String
Function ReplaceString(txt As String, from_str As String, to_str As String) As String
    Var Res As String
    Var ePos As Integer

    Res = ""
    Do
        ePos = InStr(txt, from_str)
        If ePos = 0 Then Exit do

        Res = Res & Left$(txt, ePos - 1) & to_str
        txt = Mid$(txt, ePos + Len(from_str))
    Loop

    Res = Res & txt
    ReplaceString = Res
End Function

'=======================================
'= HTML変換部
'=======================================
Declare Function TextToHTML(source_Text As String) As String
Function TextToHTML(source_Text As String) As String
    Var txt As String
    Var ePos As Integer
    Var Next_line As String

    txt = ""
    txt = txt & "<TT><PRE>"
    txt = txt & "<head>"
    txt = txt & "<meta http-equiv=" & Chr$(34) & "Content-Language" & Chr$(34) & "
content=" & Chr$(34) & "ja" & Chr$(34) & ">"
    txt = txt & "<meta http-equiv=" & Chr$(34) & "Content-Type" & Chr$(34) & " content=" &
Chr$(34) & "Text/html; charset=shift_jis" & Chr$(34) & ">"
    txt = txt & "</head>"
    txt = txt & "<font face=" & Chr$(34) & "MS ゴシック" & Chr$(34) & ">"

    Do While Len(source_Text) > 0
        ePos = instr(source_Text, vbCrLf)
        If ePos = 0 Then
            Next_line = source_Text
            source_Text = ""
        Else
            Next_line = Left$(source_Text, ePos - 1)
```

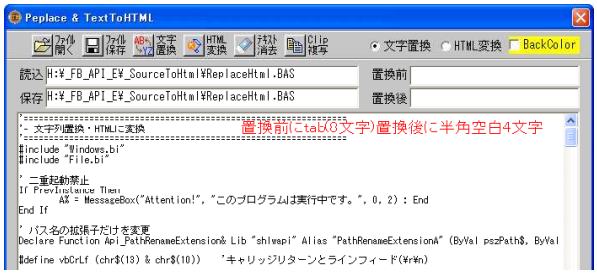
```
            source_Text = Mid$(source_Text, ePos + Len(vbCrLf))
        End if

        Next_line = ReplaceString(Next_line, "&", "&amp;")
        Next_line = ReplaceString(Next_line, "<", "&lt;")
        Next_line = ReplaceString(Next_line, ">", "&gt;")

        Next_line = Comments(Next_line)
        Next_line = ReplaceString(Next_line, """", "&quot;")

        txt = txt & Next_line & vbCrLf
    Loop

    txt = Left$(txt, Len(txt) - Len(vbCrLf))
    txt = txt & "</PRE></TT></font>" & vbCrLf

    TextToHTML = txt
End Function

'=====================================
'= EditBoxｸﾘｱ
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    For i = 0 To 3
        Edit(i).SetWindowText ""
    Next
End Sub

'=====================================
'= ﾌｧｲﾙ読込
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var hFile As byte
    Var Buff As String
    Var Temp As String
    Var Ret As Long

    Button1_on

    FileName = WinOpenDlg("ﾌｧｲﾙのｵｰﾌﾟﾝ", "*.txt;*.bas;*.rc", "ﾃｷｽﾄ(*.txt);basﾌｧｲﾙ
(*.bas);rcﾌｧｲﾙ(*.rc)", 0)
    If FileName <> Chr$(&H1B) Then
        Edit(2).SetWindowText FileName
        hFile = FreeFile
        Open FileName For BinInp As hFile
        Buff = Space$(Lof(hFile))
        FRead hFile, Buff
        Buff = JConv$(Buff, 0, 2)
        Edit(0).SetWindowText Buff
        Close hFile

        '拡張子をhtmに変換
        Temp = FileName & String$(260, Chr$(0))
        Ret = Api_PathRenameExtension(Temp, ".htm")
        Edit(3).SetWindowText Temp
    Else
        FileName = ""
    End If
End Sub

'=====================================
'= ソース → HTML変換
'=====================================
Declare Sub Button3_on edecl ()
Sub Button3_on()
    If Edit(0).GetWindowText = "" Then Exit Sub
    Edit(1).SetWindowText TextToHTML(Edit(0).GetWindowText)
End Sub
```

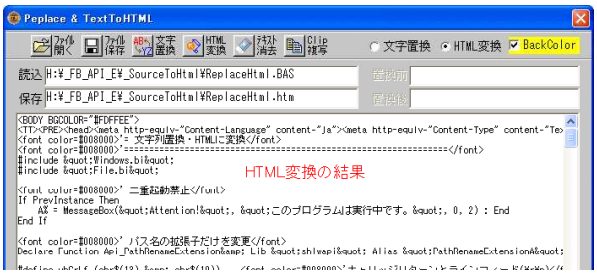
```
'=====================================
'= 変換済データをHTMLとして保存
'=====================================
Declare Sub Button4_on edecl ()
Sub Button4_on()
    Var hFile As byte
    Var Buff As String

    FileName = Edit(3).GetWindowText

    If Edit(1).GetWindowText = "" Then
        Exit Sub
    Else If FileName = "" Then
        FileName = "default.htm"
        Edit(3).SetWindowText FileName
    End if

    If IsExistFile(FileName) Then Kill FileName
    hFile = FreeFile
    Open FileName For BinIO As hFile
    Buff = Edit(1).GetWindowText
    FWrite hFile, Buff
    Close hFile
End Sub

'=====================================
'= リサイズ
'=====================================
Declare Sub MainForm_Resize edecl ()
Sub MainForm_Resize()

    '最小基本サイズに戻す
    If GetWidth < 700 Or GetHeight < 512 Then
        SetWindowSize 700, 512
    End if

    Text(0).MoveWindow 12, 16
    Text(1).MoveWindow 12, (GetHeight - 78) / 2 + 15

    Edit(0).MoveWindow 10, 36
    Edit(0).SetWindowSize GetWidth - 34, (GetHeight - 78) / 2 - 37

    Edit(1).MoveWindow 10, (GetHeight - 78) / 2 + 11 + 26
    Edit(1).SetWindowSize GetWidth - 34, (GetHeight - 78) / 2 - 37

    Edit(2).MoveWindow 96, 10
    Edit(2).SetWindowSize GetWidth - 120, 22

    Edit(3).MoveWindow 96, (GetHeight - 78) / 2 + 11
    Edit(3).SetWindowSize GetWidth - 120, 22

    Button(0).MoveWindow  38, GetHeight - 66
    Button(1).MoveWindow 194, GetHeight - 66
    Button(2).MoveWindow 350, GetHeight - 66
    Button(3).MoveWindow 506, GetHeight - 66

    'サイズグリップ(右下)
    Text(2).MoveWindow (GetWidth - Text(2).GetWidth) - zX, GetHeight - Text(2).GetHeight
- zY
End Sub

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

HTML変換と文字列置換

PathRenameExtension パス名の拡張子だけを変更

F-Basicで作成したbasファイルは、FrontPageでhtmlに変換しているのですが、Tabを8文字から4文字に、コメント部分は暗緑色にとか背景色をFDFFFFに統一したいとかいろいろ欲が出てきましたので、HTML変換と置換関数を一つにしました。
通常、「ファイル保存」をクリックするとHTMLの場合.htm、置換の場合は読込と同じファイル名で保存します。
FrontPageのHTMLに貼り付けるため、「Clip複写」を設けています。
「ファイルを開く」、または目的ファイルを「読み込み」EditBoxへドラッグ＆ドロップします。






```
'=======================================
'= 文字列置換･HTMLに変換
'=    (ReplaceHtml.bas)
'=======================================
#include "Windows.bi"
#include "File.bi"

' 二重起動禁止
If PrevInstance Then
    A% = MessageBox("Attention!", "このプログラムは実行中です。", 0, 2) : End
End If

' パス名の拡張子だけを変更
Declare Function Api_PathRenameExtension& Lib "shlwapi" Alias "PathRenameExtensionA"
(ByVal pszPath$, ByVal pszExt$)

#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 Edit(4) As object
Var Shared Text(9) As Object
Var Shared BmpButton(5) As Object
Var Shared Radio(1) As Object
Var Shared Check1 As Object

For i = 0 To 4
    Edit(i).Attach GetDlgItem("Edit" & Trim$(Str$(i + 1)))
    If i < 2 Or i > 2 Then
        Edit(i).SetFontSize 14
    Else
        Edit(i).SetFontSize 12
    End If
Next
For i = 0 To 9
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1)))
    If i < 5 Or i = 9 Then
        Text(i).SetFontSize 12
    Else
        Text(i).SetFontSize 14
    End If
Next
For i = 0 To 5
    BmpButton(i).Attach GetDlgItem("BmpButton" & Trim$(Str$(i + 1)))
Next
For i = 0 To 1
    Radio(i).Attach GetDlgItem("Radio" & Trim$(Str$(i + 1)))
    Radio(i).SetFontSize 14
Next
Check1.Attach GetDlgItem("Check1") : Check1.SetFontSize 14
```

```
Var Shared FileName As String                'ファイル名
Var Shared zX As Integer                     'サイズグリップX軸
Var Shared zY As Integer                     'サイズグリップY軸
Var Shared FLG As Integer                    '置換/HTML変換
Var Shared Buffer As String

'=====================================
'= 拡張子をhtmに変換
'=====================================
Declare Sub ExtensionSet()
Sub ExtensionSet()
    Var Temp As String
    Var Ret As Long

    '拡張子をhtmに変換
    Temp = FileName & String$(260, Chr$(0))
    Ret = Api_PathRenameExtension(Temp, ".htm")
    Edit(1).SetWindowText Temp
End Sub

'=====================================
'= 注釈を暗緑色で
'=====================================
Declare Function Comments(txt As String) As String
Function Comments(txt As String) As String
    Var ePos As Integer
    Var ch As String
    Var quote As Integer

    For ePos = 1 To Len(txt)
        ch = Mid$(txt, ePos, 1)

        If ch = """" Then
            quote = Not quote
        Else If (ch = "'") And (Not quote) Then
            Exit For
        End If
    Next ePos

    If ePos <= Len(txt) Then

        '暗緑色で表示
        Comments = Left$(txt, ePos - 1) & "<font color=" & "#008000" & ">'" & Mid$(txt, ePos
+ 1) & "</font>"

        '太字で表示する場合
        'Comments = Left$(txt, ePos - 1) & "<B>'" & Mid$(txt, ePos + 1) & "</B>"
    Else
        Comments = txt
    End If
End Function

'=====================================
'= 置換関数
'= sOriginal:原本 sCorrection:修正文字 sReplace:置換文字
'=====================================
Declare Function Replace(strOrgriginal As String, strCororrection As String, strReplace
As String) As String
Function Replace(strOrgriginal As String, strCororrection As String, strReplace As
String) As String
    Var txt As String
    Var ePos As Integer
    txt = ""
    Do
        ePos = InStr(strOrgriginal, strCororrection)
        If ePos = 0 Then Exit Do

        txt = txt & Left$(strOrgriginal, ePos - 1) & strReplace
        strOrgriginal = Mid$(strOrgriginal, ePos + Len(strCororrection))
    Loop
```

```
    txt = txt & strOrgriginal
    Replace = txt
End Function

'=====================================
'= HTML変換部
'=====================================
Declare Function TextToHTML(SourceText As String) As String
Function TextToHTML(SourceText As String) As String
    Var txt As String
    Var ePos As Integer
    Var NextLine As String

    txt = ""

    'チェック無しの場合(BodyColor=無)
    If Check1.GetCheck = 0 Then
        txt = ""

    'チェック有りの場合(BodyColor=#FDFFEE)
    Else
        txt = txt & "<BODY BGCOLOR=" & Chr$(34) & "#FDFFEE" & Chr$(34) & ">" & vbCrLf
    End If

    txt = txt & "<TT><PRE>"
    txt = txt & "<head>"
    txt = txt & "<meta http-equiv=" & Chr$(34) & "Content-Language" & Chr$(34) & "
content=" & Chr$(34) & "ja" & Chr$(34) & ">"
    txt = txt & "<meta http-equiv=" & Chr$(34) & "Content-Type" & Chr$(34) & " content=" &
Chr$(34) & "Text/html; charset=shift_jis" & Chr$(34) & ">"
    txt = txt & "</head>"
    txt = txt & "<font face=" & Chr$(34) & "MS ゴシック" & Chr$(34) & ">"

    Do While Len(SourceText) > 0
        ePos = InStr(SourceText, vbCrLf)
        If ePos = 0 Then
            NextLine = SourceText
            SourceText = ""
        Else
            NextLine = Left$(SourceText, ePos - 1)
            SourceText = Mid$(SourceText, ePos + Len(vbCrLf))
        End If

        NextLine = Replace(NextLine, "&", "&amp;")
        NextLine = Replace(NextLine, "<", "&lt;")
        NextLine = Replace(NextLine, ">", "&gt;")

        NextLine = Comments(NextLine)
        NextLine = Replace(NextLine, """", "&quot;")

        txt = txt & NextLine & vbCrLf
    Loop

    txt = Left$(txt, Len(txt) - Len(vbCrLf))
    txt = txt & "</PRE></TT></font>" & vbCrLf

    TextToHTML = txt
End Function

'=====================================
'= 文字列置換を選択
'=====================================
Declare Sub Radio1_on edecl ()
Sub Radio1_on()
    FLG = 0

    Text(7).EnableWindow -1
    Text(8).EnableWindow -1
    Edit(3).EnableWindow -1
    Edit(4).EnableWindow -1
```

```
    Edit(1).SetWindowText Edit(0).GetWindowText
End Sub

'=====================================
'= HTML変換を選択
'=====================================
Declare Sub Radio2_on edecl ()
Sub Radio2_on()
    FLG = 1

    Edit(3).SetWindowText ""
    Edit(4).SetWindowText ""

    Text(7).EnableWindow 0
    Text(8).EnableWindow 0
    Edit(3).EnableWindow 0
    Edit(4).EnableWindow 0

    ExtensionSet
End Sub

'=====================================
'= EditBoxクリア
'=====================================
Declare Sub BmpButton5_on edecl ()
Sub BmpButton5_on()
    For i = 0 To 4
        Edit(i).SetWindowText ""
    Next
    SetFocus
End Sub

'=====================================
'= ファイル読込
'=====================================
Declare Sub FileRead()
Sub FileRead()
    Var hFile As Byte
    Var Temp As String
    Var Ret As long

    On Error GoTo *ErTrap
    BmpButton5_on                       'Edit3消去連動
    Radio1_on                           '
    Radio(0).SetCheck 1
    Radio(1).SetCheck 0
    Check1.SetCheck 0

    Edit(0).SetWindowText FileName
    hFile = FreeFile

    Open FileName For BinInp As hFile
        Buffer = Space$(lof(hFile))
        fread hFile, Buffer
        Buffer = jconv$(Buffer, 0, 2)
        Edit(2).SetWindowText Buffer
    Close hFile

    If FLG = 0 Then
        Edit(1).SetWindowText FileName
    Else
        ExtensionSet
    End If
    Exit Sub

*ErTrap
    Close hFile
    Edit(0).SetWIndowText ""
    Resume *ErReturn
```

```
*ErReturn
    On Error GoTo 0
End Sub

'=====================================
'= ファイル読込
'=====================================
Declare Sub BmpButton1_on edecl ()
Sub BmpButton1_on()
    FileName = WinOpEndlg("ファイルのオープン", "*.txt;*.bas;*.rc", "ﾃｷｽﾄ(*.txt);basﾌｧｲﾙ
(*.bas);rcﾌｧｲﾙ(*.rc)", 0)
    If FileName => Chr$(&H1B) Then
        FileRead
    Else
        FileName = ""
    End If
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)
    Edit(0).SetWindowText FileName
    Edit(1).SetWindowText FileName
    FileRead
End Sub

'=====================================
'= シェルドロップされたファイル名を取得
'=====================================
Declare Sub MainForm_DropFiles edecl (ByVal DF As Long)
Sub MainForm_DropFiles(ByVal DF As Long)
    Var CN As Long
    CN = GetDropFileCount(DF)
    FileName = GetDropFileName(DF, 0)
    Edit(0).SetWindowText FileName
    Edit(1).SetWindowText FileName
    FileRead
End Sub

'=====================================
'= FLG=0(同名で上書保存)･FLG=1(HTMLとして保存)
'=====================================
Declare Sub BmpButton2_on edecl ()
Sub BmpButton2_on()
    Var hFile As Byte

    FileName = Edit(0).GetWindowText
    If Edit(2).GetWindowText = "" Then
        Exit Sub                            'Buffer(Edit(2))に何も入っていない場合
    Else If FileName = "" Then
        FileName = "default.htm"            'ファイル名が無い場合(デフォルト名)
        Edit(1).SetWindowText FileName
    End If

    FileName = Edit(1).GetWindowText        'ファイル名はEdit(1)より取得

    If IsExistFile(FileName) Then kill FileName
    hFile = FreeFile

    Open FileName For BinIo As hFile
        Buffer = Edit(2).GetWindowText
        fwrite hFile, Buffer
    Close hFile
```

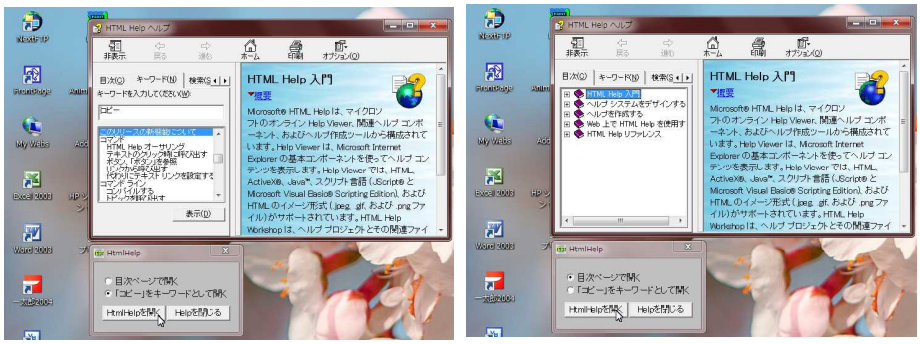
```
    SetFocus
End Sub

'=====================================
'= 文字列置換実行
'=====================================
Declare Sub BmpButton3_on edecl ()
Sub BmpButton3_on()
    If Edit(3).GetWindowText = "" Then
        A% = MessageBox("Attention!", "検索文字列を指定してください！", 0, 2)
        Exit Sub
    End If

    Var strOrg As String
    Var strCor As String
    Var strRep As String

    strCor = Edit(3).GetWindowtext          '検索文字列
    strRep = Edit(4).GetWindowtext          '置換後文字列
    strOrg = Edit(2).GetWindowtext          'バッファ内の文字列

    '----------------------------------------
    Buffer = Replace(strOrg, strCor, strRep)
    '----------------------------------------

    EDit(2).SetWindowText Buffer
    SetFocus
End Sub

'=====================================
'= ソース → HTML変換
'=====================================
Declare Sub BMpButton4_on edecl ()
Sub BmpButton4_on()
    If Edit(2).GetWindowText = "" Then SetFocus : Exit Sub

    ExtensionSet
    Radio2_on
    Radio(0).SetCheck 0
    Radio(1).SetCheck 1

    If Edit(2).GetWindowText = "" Then
        SetFocus : Exit Sub
    Else if Ucase$(Left$(Edit(2).GetWindowText, 9)) = "<TT><PRE>" Or
Ucase$(Left$(Edit(2).GetWindowText, 9)) = "<BODY BGC" Or
Ucase$(Left$(Edit(2).GetWindowText, 6)) = "<HTML>" Then
        A% = MessageBox(GetWindowText, "既に変換されています！",0, 2)
        SetFocus : Exit Sub
    End If

    Edit(2).SetWindowText TextToHTML(Edit(2).GetWindowText)
    SetFocus
End Sub

'=====================================
'= コピー
'=====================================
Declare Sub BmpButton6_on edecl ()
Sub BmpButton6_on()
    Edit(2).SetSelText 0, -1
    Edit(2).Copy
    Edit(2).SetSelText -1, 0
    SetFocus
End sub

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

```
    Line(0, 1) - (GetWidth, 1),, 1 : Line(0, 2) - (GetWidth, 2),, 15
    Line(0, 33) - (GetWidth, 33),, 1 : Line(0, 34) - (GetWidth, 34),, 15
End Sub

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

HTMLヘルプを開く

HTMLヘルプを開きます。
HtmlHelp HTMLヘルプウィンドウを表示する関数
HtmlHelpTopic 検索語でHTMLヘルプを開く

例では、HTML Help 入門を開いています。



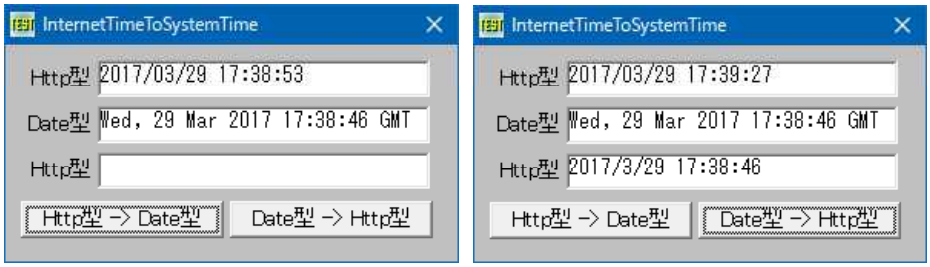
```
'=======================================
'=  HTMLヘルプを開く
'=    (Htmlhelp.bas)
'=======================================
#include "Windows.bi"

#define HH_DISPLAY_TOPIC &H0                'ヘルプトピックを開く
#define HH_DISPLAY_TOC &H1                  '[目次]タブを選択
#define HH_DISPLAY_INDEX &H2                'キーワード表示
#define HH_DISPLAY_SEARCH &H3               '[検索]タブを選択
#define HH_DISPLAY_TEXT_POPUP &HE           '明示的テキスト文字列、リソースIDに基づいたテキスト文字列などを開く

#define HH_SET_WIN_TYPE &H4                 'ヘルプウィンドウの実行時に、新しいヘルプ ウィンドウを作成するか、または既存のヘルプウィンドウを変更

#define HH_GET_WIN_TYPE &H5                 '指定されたウィンドウタイプに関連付けられたHH_WINTYPE構造体へのポインタを取得

#define HH_GET_WIN_HANDLE &H6               '指定されたウィンドウタイプのウィンドウハンドルを返す
#define HH_HELP_CONTEXT &HF                 'dwData(開くページのID)に従う
#define HH_TP_HELP_CONTEXTMENU &H10         'ポップアップコンテキストメニューを開く
#define HH_TP_HELP_WM_HELP &H11             'ポップアップヘルプトピックを開く
#define HH_CLOSE_ALL &H12                   'ソフトと同時行動

' HTMLヘルプウィンドウを表示する関数
Declare Function Api_HtmlHelp& Lib "hhctrl.ocx" Alias "HtmlHelpA" (ByVal hwndCaller&,
ByVal pszFile$, ByVal uCommand&, ByVal dwData&)

' 検索語でHTMLヘルプを開く
Declare Function Api_HtmlHelpTopic& Lib "hhctrl.ocx" Alias "HtmlHelpA" (ByVal hWnd&,
ByVal lpHelpFile$, ByVal wCommand&, ByVal dwData$)

Declare Sub HelpClose()
```

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

'====================================
'= HTMLヘルプ入門を開く
'====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var FileName As String
    Var Ret As Long

    FileName = "c:¥Windows¥help¥htmlhelp.chm"

    'ヘルプウィンドウのハンドル
    If Index = 0 Then
        Ret = Api_HtmlHelp(GethWnd, FileName, HH_DISPLAY_TOC, 0)
    Else
        Ret = Api_HtmlHelpTopic(GethWnd, FileName, HH_DISPLAY_INDEX, "コピー")
    End If
End Sub

'====================================
'=
'====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    HelpClose
End Sub

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

    HelpClose
    If Cancel = 0 Then End
End Sub

'====================================
'=
'====================================
Sub HelpClose()
    Var Ret As Long

    Ret = Api_HtmlHelp(GethWnd, "", HH_CLOSE_ALL, 0)
End Sub

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

HTTP形式・SYSTEMTIME形式時間の相互変換

インターネット形式とDate形式の時間を相互に変換します。
InternetTimeToSystemTime インターネット(HTTP)形式の時間をSYSTEMTIME形式時間に変換

InternetTimeFromSystemTime SYSTEMTIME形式の時間をインターネット(HTTP)形式時間に変換

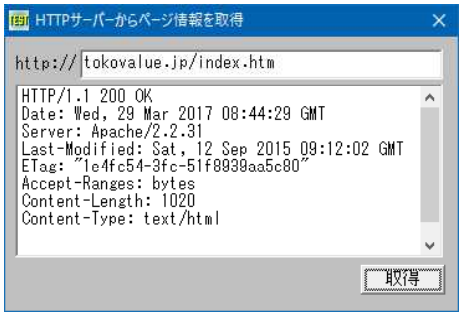



```
'=====================================
'= HTTP形式･SYSTEMTIME形式時間の相互変換
'=    (InternetTimeToSystemTime.bas)
'=====================================
#include "Windows.bi"

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

' インターネット(HTTP)形式の時間をSYSTEMTIME形式時間に変換
Declare Function Api_InternetTimeToSystemTime& Lib "wininet" Alias
"InternetTimeToSystemTimeA" (ByVal lpszTime$, ByRef pst As SYSTEMTIME, ByVal
dwReserved&)

' SYSTEMTIME形式の時間をインターネット(HTTP)形式時間に変換
Declare Function Api_InternetTimeFromSystemTime& Lib "wininet" Alias
"InternetTimeFromSystemTimeA" (ByRef pst As SYSTEMTIME, ByVal dwRFC&, ByVal lpszTime$,
ByVal cbTime&)
#define INTERNET_RFC1123_FORMAT 0          'RFC1123形式
#define INTERNET_RFC1123_BUFSIZE 30               '

Var Shared Text(5) As Object
Var Shared Button(1) As Object
Var Shared Timer1 As Object

For i = 0 To 5
    If i < 2 Then
        Button(i).Attach GetDlgItem("Button" & Trim$(Str$(i + 1)))
        Button(i).SetFontSize 14
    End If
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1))) : Text(i).SetFontSize 14
Next
Timer1.Attach GetDlgItem("Timer1")

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Text(0).SetWindowtext Today$ & " " & Time$
    Timer1.SetInterval 10
    Timer1.Enable -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Text(0).SetWindowtext Today$ & " " & Time$
```

```
End Sub

'=====================================
'= HTTP --> DATE
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var dt As String
    Var st As SYSTEMTIME
    Var strTime As String

    dt = Text(0).GetWindowText

    st.wYear = Val(Left$(dt, 4))
    st.wMonth = Val(Mid$(dt, 6, 2))
    st.wDay = Val(Mid$(dt, 9, 2))
    st.wHour = Val(Mid$(dt, 12, 2))
    st.wMinute = Val(Mid$(dt, 15, 2))
    st.wSecond = Val(Right$(dt, 2))

    strTime = String$(INTERNET_RFC1123_BUFSIZE + 1, 0)

    Ret = Api_InternetTimeFromSystemTime(st, INTERNET_RFC1123_FORMAT, strTime,
Len(strTime))

    Text(1).SetWindowText strTime
    Text(2).SetWindowText ""
End Sub

'=====================================
'= DATE --> HTTP
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var strTime As String
    Var st As SYSTEMTIME
    Var Ret As Long
    strTime =Text(1).GetWindowText

    Ret = Api_InternetTimeToSystemTime(strTime, st, 0)

    Text(2).SetWindowText Trim$(Str$(st.wYear)) & "/" & Trim$(Str$(st.wMonth)) & "/" &
Trim$(Str$(st.wDay)) & " " & Trim$(Str$(st.wHour)) & ":" & Trim$(Str$(st.wMinute)) & ":" &
Trim$(Str$(st.wSecond))
End Sub

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

HTTPサーバーからページ情報を取得

InternetOpen インターネットのハンドルを作成
InternetConnect インターネット上のサーバーに接続し、ハンドルを返す
HttpOpenRequest HTTPリクエストの作成
HttpSendRequest 指定されたHTTPリクエストをサーバーへ送信
HttpQueryInfo HTTPリクエストに関する情報を取得
InternetCloseHandle Win32インターネット関数のハンドルをクローズ
InternetQueryOption 指定されたハンドルに対するインターネットオプション情報を取得



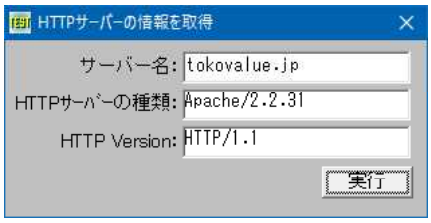

```
'=====================================
'= HTTPサーバーからページ情報を取得
'=    (HttpOpenRequest.bas)
'=====================================
#include "Windows.bi"

' インターネットのハンドルの作成
Declare Function Api_InternetOpen& Lib "wininet" Alias "InternetOpenA" (ByVal sAgent$,
ByVal lAccessType&, ByVal sProxyName$, ByVal sProxyBypass$, ByVal lFlags&)

' インターネット上のサーバに接続し、ハンドルを返す
Declare Function Api_InternetConnect& Lib "wininet" Alias "InternetConnectA" (ByVal
hSession&, ByVal sServerName$, ByVal nServerPort&, ByVal sUsername$, ByVal sPassword$,
ByVal lService&, ByVal lFlags&, ByVal lContext&)

' HTTPリクエストの作成
Declare Function Api_HttpOpenRequest& Lib "wininet" Alias "HttpOpenRequestA" (ByVal
hHttpSession&, ByVal sVerb$, ByVal sObjectName$, ByVal sVersion$, ByVal sReferer$, ByVal
something&, ByVal lFlags&, ByVal lContext&)

' 指定されたHTTPリクエストをサーバへ送信
Declare Function Api_HttpSendRequest& Lib "wininet" Alias "HttpSendRequestA" (ByVal
HttpRequest&, ByVal sHeaders$, ByVal lHeadersLength&, sOptional As Any, ByVal
lOptionalLength&)

' HTTPリクエストに関連する情報の取得
Declare Function Api_HttpQueryInfo& Lib "wininet" Alias "HttpQueryInfoA" (ByVal
HttpRequest&, ByVal lInfoLevel&, ByRef Buffer As Any, ByRef BufferLen&, ByRef lIndex&)

' Win32インターネット関数のハンドルのクローズ
Declare Function Api_InternetCloseHandle& Lib "wininet" Alias "InternetCloseHandle"
(ByVal hInet&)

' 指定されたハンドルに対するインターネットのオプション情報を取得
Declare Function Api_InternetQueryOption& Lib "wininet" Alias "InternetQueryOptionA"
(ByVal hInternet&, ByVal lOption&, ByRef Buffer As Any, ByRef BufferLen&)

#define INTERNET_OPEN_TYPE_PRECONFIG 0     'プロキシまたはレジストリから直接コンフィギュレーションを
                                           取得
#define INTERNET_DEFAULT_FTP_PORT 21       'RFCで既定されたftpポート
#define INTERNET_DEFAULT_HTTP_PORT 80      'RFCで既定されたhttpポート
#define INTERNET_DEFAULT_HTTPS_PORT 443    'RFCで既定されたhttpsポート
#define INTERNET_SERVICE_HTTP 3            'HTTPサービス
#define INTERNET_FLAG_RELOAD -2147483648   'ダウンロード時キャッシュを使わずサーバーから行う
#define HTTP_QUERY_STATUS_CODE 19          'サーバから返された状態コード
#define HTTP_QUERY_STATUS_TEXT 20          'サーバから返された補足のテキスト
#define HTTP_QUERY_RAW_HEADERS 21          'NUL文字で区切られた全てのヘッダ
#define HTTP_QUERY_RAW_HEADERS_CRLF 22     'CR/LFで区切られた全てのヘッダ
#define HTTP_QUERY_FLAG_REQUEST_HEADERS -2147483648 '応答ヘッダの取得
#define UserAgent "http sample"
Var Shared hSession As Long
Var Shared hConnect As Long
Var Shared hHttpOpenRequest As Long

Var Shared Edit1 As Object
Var Shared Edit2 As Object
Var Shared Button1 As Object
```

```
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Edit2.Attach GetDlgItem("Edit2") : Edit2.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Function GetQueryInfo(ByVal HttpRequest As Long, ByVal InfoLevel As Long) As
String
Function GetQueryInfo(ByVal HttpRequest As Long, ByVal InfoLevel As Long) As String
    Var Buffer As String * 1024
    Var BufferLen As Long
    Var Ret As Long

    BufferLen = Len(Buffer)
    Ret = Api_HttpQueryInfo(HttpRequest, InfoLevel, Buffer, BufferLen, 0)
    GetQueryInfo = Left$(Buffer, BufferLen)
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Host As String
    Var StatusCode As String
    Var StatusText As String
    Var RawHeaders As String
    Var RawHeadersCrLf As String
    Var Server As String
    Var Ret As Long

    Edit2.SetWindowText ""

    Host = Left$(Edit1.GetWindowText, InStr(Edit1.GetWindowText, "/") - 1)

    'インターネットに接続       http sample  レジストリの設定に従う
    hSession = Api_InternetOpen(UserAgent, INTERNET_OPEN_TYPE_PRECONFIG, ByVal 0, ByVal
0, 0)

    If CLng(hSession) Then

        'HTTPサーバーに接続
        hConnect = Api_InternetConnect(hSession, Host, INTERNET_DEFAULT_HTTP_PORT, ByVal
0, ByVal 0, INTERNET_SERVICE_HTTP, 0, 0)

        If hConnect > 0 Then

            'サーバー上で欲しいURLを指定
            hHttpOpenRequest = Api_HttpOpenRequest(hConnect, "HEAD",
Right$(Edit1.GetWindowText, Len(Edit1.GetWindowText) - InStr(Edit1.GetWindowText, "/")
+ 1), "HTTP/1.1", ByVal 0, 0, INTERNET_FLAG_RELOAD, 0)

            If Clng(hHttpOpenRequest) Then

                'ヘッダーを要求
                Ret = Api_HttpSendRequest(hHttpOpenRequest, ByVal 0, 0, 0, 0)

                If Ret Then                    'ファイルの内容を受信

                    'サーバから返された状態コード
                    StatusCode = GetQueryInfo(hHttpOpenRequest, HTTP_QUERY_STATUS_CODE)

                    'サーバから返された補足のテキスト
                    StatusText = GetQueryInfo(hHttpOpenRequest, HTTP_QUERY_STATUS_TEXT)

                    'NUL文字で区切られた全てのヘッダ
                    RawHeaders = GetQueryInfo(hHttpOpenRequest, HTTP_QUERY_RAW_HEADERS)
```

```
                    'CR/LFで区切られた全てのヘッダ
                    RawHeadersCrLf = GetQueryInfo(hHttpOpenRequest,
HTTP_QUERY_RAW_HEADERS_CRLF)
                    Edit2.SetWindowText RawHeadersCrLf
                End If
            End If
        End If
    End If

    'すべてのハンドルをクローズ
    Ret = Api_InternetCloseHandle(hHttpOpenRequest)
    Ret = Api_InternetCloseHandle(hSession)
    Ret = Api_InternetCloseHandle(hConnect)
End Sub

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

HTTPサーバーの情報を取得

InternetOpen インターネットのハンドルを作成
InternetConnect インターネット上のサーバーに接続し、ハンドルを返す
HttpOpenRequest HTTPリクエストの作成
HttpSendRequest 指定されたHTTPリクエストをサーバーへ送信
HttpQueryInfo HTTPリクエストに関する情報を取得
InternetCloseHandle Win32インターネット関数のハンドルをクローズ



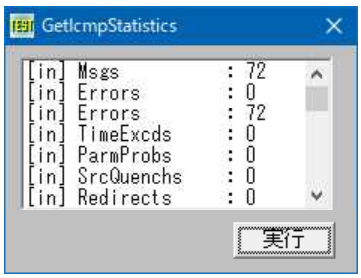

```
'=======================================
'= HTTPサーバーの情報を取得
'=    (HttpSendRequest.bas)
'=======================================
#include "Windows.bi"

' インターネットのハンドルの作成
Declare Function Api_InternetOpen& Lib "wininet" Alias "InternetOpenA" (ByVal sAgent$,
ByVal lAccessType&, ByVal sProxyName$, ByVal sProxyBypass$, ByVal lFlags&)

' インターネット上のサーバに接続し、ハンドルを返す
Declare Function Api_InternetConnect& Lib "wininet" Alias "InternetConnectA" (ByVal
hSession&, ByVal sServerName$, ByVal nServerPort&, ByVal sUsername$, ByVal sPassword$,
ByVal lService&, ByVal lFlags&, ByVal lContext&)

' HTTPリクエストの作成
Declare Function Api_HttpOpenRequest& Lib "wininet" Alias "HttpOpenRequestA" (ByVal
hHttpSession&, ByVal sVerb$, ByVal sObjectName$, ByVal sVersion$, ByVal sReferer$, ByVal
something&, ByVal lFlags&, ByVal lContext&)

' 指定されたHTTPリクエストをサーバへ送信
Declare Function Api_HttpSendRequest& Lib "wininet" Alias "HttpSendRequestA" (ByVal
HttpRequest&, ByVal sHeaders$, ByVal lHeadersLength&, sOptional As Any, ByVal
lOptionalLength&)
```

```
' HTTPリクエストに関連する情報の取得
Declare Function Api_HttpQueryInfo& Lib "wininet" Alias "HttpQueryInfoA" (ByVal
HttpRequest&, ByVal lInfoLevel&, ByRef Buffer As Any, ByRef BufferLen&, ByRef lIndex&)

' Win32インターネット関数のハンドルのクローズ
Declare Function Api_InternetCloseHandle% Lib "wininet" Alias "InternetCloseHandle"
(ByVal hInet&)

#define INTERNET_OPEN_TYPE_PRECONFIG 0     'IEの設定に従い接続
#define INTERNET_SERVICE_HTTP 3            'HTTP・HTTPSを指定
#define INTERNET_FLAG_RELOAD -2147483648   'ローカルのキャッシュを無視し、常にサーバからデータ取得
#define INTERNET_DEFAULT_HTTP_PORT 80      'RFCで既定されたhttpポート
#define HTTP_QUERY_CONTENT_TYPE 1          'リソースのタイプ
#define HTTP_QUERY_VERSION 18              'サーバに最後に返された応答コード
#define HTTP_QUERY_SERVER 37               '
#define HTTP_QUERY_TITLE 38                '

Var Shared Text(4) As Object
Var Shared Edit1 As Object
Var Shared Button1 As Object

For i = 0 To 4
    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 Button1_on edecl ()
Sub Button1_on()
    Var hInet As Long
    Var hConnect As Long
    Var hRequest As Long
    Var Ret As Long
    Var Buffer As String * 1024

    '初期化
    hInet = Api_InternetOpen("", INTERNET_OPEN_TYPE_PRECONFIG, ByVal 0, ByVal 0, 0)

    If hInet <> 0 Then

        'HTTPサービスに接続
       hConnect = Api_InternetConnect(hInet, Edit1.GetWindowText,
INTERNET_DEFAULT_HTTP_PORT, ByVal 0, ByVal 0, INTERNET_SERVICE_HTTP, 0, 0)

       If hConnect <> 0 Then

           'HTTPリクエストのハンドルを開く
           hRequest = Api_HttpOpenRequest(hConnect, "GET", ByVal 0, ByVal 0, ByVal 0, 0,
INTERNET_FLAG_RELOAD, 0)

           If hRequest <> 0 Then

               'HTTPサーバーにリクエストを送信
               Ret = Api_HttpSendRequest(hRequest, ByVal 0, 0, ByVal 0, 0)

               If Ret Then

                   'HTTPサーバーの種類をリクエスト
                   If Api_HttpQueryInfo(hRequest, HTTP_QUERY_SERVER, Buffer,
Len(Buffer), 0) <> 0 Then
                       Text(3).SetWindowtext Buffer
                   End If

                   'HTTPのバージョンをリクエスト
                   If Api_HttpQueryInfo(hRequest, HTTP_QUERY_VERSION, Buffer,
Len(Buffer), 0) <> 0 Then
```

```
                    Text(4).SetWindowText Buffer
                End If
            End If
        End If
    End If

    ' ハンドルを閉じる
    If hInet <> 0 Then
        Ret = Api_InternetCloseHandle(hInet)
    End If

    If hConnect <> 0 Then
        Ret = Api_InternetCloseHandle(hConnect)
    End If

    If hRequest <> 0 Then
        Ret = Api_InternetCloseHandle(hRequest)
    End If
End Sub

'=======================================
'=
'=======================================
Declare Sub Edit1_SetFocus edecl ()
Sub Edit1_SetFocus()
    Text(3).SetWindowText ""
    Text(4).SetWindowtext ""
End Sub

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

ICMP統計値を取得

GetIcmpStatistics ICMP(Internet Control Message Protocol)統計値を取得




```
'=======================================
'= ICMP統計値を取得
'=    (GetIcmpStatistics.bas)
'=======================================
#include "Windows.bi"

Type MIBICMPSTATS
    dwMsgs          As Long
    dwErrors        As Long
    dwDestUnreachs  As Long
    dwTimeExcds     As Long
    dwParmProbs     As Long
    dwSrcQuenchs    As Long
```

```
    dwRedirects       As Long
    dwEchos           As Long
    dwEchoReps        As Long
    dwTimestamps      As Long
    dwTimestampReps   As Long
    dwAddrMasks       As Long
    dwAddrMaskReps    As Long
End Type

Type MIBICMPINFO
    icmpInStats       As MIBICMPSTATS
    icmpOutStats      As MIBICMPSTATS
End Type

Type MIB_ICMP
    stats      As MIBICMPINFO
End Type

' ICMP(Internet Control Message Protocol)統計値を取得
Declare Function Api_GetIcmpStatistics& Lib "iphlpapi" Alias "GetIcmpStatistics" (pStats
As MIB_ICMP)

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 mi As MIB_ICMP
    Var Item As String
    Var Ret As Long
    List1.Resetcontent

    'ICMP統計値を取得
    Ret = Api_GetIcmpStatistics(mi)

    '[in] Msgsを表示
    Item = Trim$(Str$(mi.stats.icmpInStats.dwMsgs))
    List1.AddString "[in] Msgs         : " & Item

    '[in] Errorsを表示
    Item = Trim$(Str$(mi.stats .icmpInStats.dwErrors))
    List1.AddString "[in] Errors       : " & Item

    '[in] DestUnreachsを表示
    Item = Trim$(Str$(mi.stats.icmpInStats.dwDestUnreachs))
    List1.AddString "[in] Errors       : " & Item

    '[in] TimeExcdsを表示
    Item = Trim$(Str$(mi.stats.icmpInStats.dwTimeExcds))
    List1.AddString "[in] TimeExcds    : " & Item

    '[in] ParmProbsを表示
    Item = Trim$(Str$(mi.stats.icmpInStats.dwParmProbs))
    List1.AddString "[in] ParmProbs    : " & Item

    '[in] SrcQuenchsを表示
    Item = Trim$(Str$(mi.stats.icmpInStats.dwSrcQuenchs))
    List1.AddString "[in] SrcQuenchs   : " & Item

    '[in] Redirectsを表示
    Item = Trim$(Str$(mi.stats.icmpInStats.dwRedirects))
    List1.AddString "[in] Redirects    : " & Item

    '[in] Echosを表示
    Item = Trim$(Str$(mi.stats.icmpInStats.dwEchos))
```

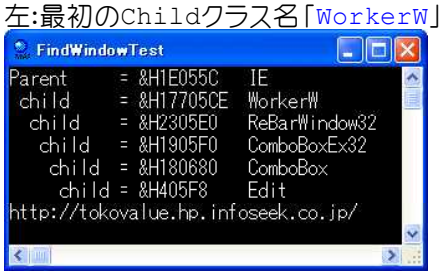
```
List1.AddString "[in] Echos        : " & Item

'[in] EchoRepsを表示
Item = Trim$(Str$(mi.stats.icmpInStats.dwEchoReps))
List1.AddString "[in] EchoReps     : " & Item

'[in] Timestampsを表示
Item = Trim$(Str$(mi.stats.icmpInStats.dwTimestamps))
List1.AddString "[in] Timestamps   : " & Item

'[in] TimestampRepsを表示
Item = Trim$(Str$(mi.stats.icmpInStats.dwTimestampReps))
List1.AddString "[in] TimestampReps : " & Item

'[in] AddrMasksを表示
Item = Trim$(Str$(mi.stats.icmpInStats.dwAddrMasks))
List1.AddString "[in] AddrMasks    : " & Item

'[in] AddrMaskRepsを表示
Item = Trim$(Str$(mi.stats.icmpInStats.dwAddrMaskReps))
List1.AddString "[in] AddrMaskReps  : " & Item

'[out] Msgsを表示
Item = Trim$(Str$(mi.stats.icmpOutStats.dwMsgs))
List1.AddString "[out] Msgs        : " & Item

'[out] Errorsを表示
Item = Trim$(Str$(mi.stats.icmpOutStats.dwErrors))
List1.AddString "[out] Errors      : " & Item

'[out] DestUnreachsを表示
Item = Trim$(Str$(mi.stats.icmpOutStats.dwDestUnreachs))
List1.AddString "[out] DestUnreachs : " & Item

'[out] TimeExcdsを表示
Item = Trim$(Str$(mi.stats.icmpOutStats.dwTimeExcds))

List1.AddString "[out] TimeExcds   : " & Item
'[out] ParmProbsを表示
Item = Trim$(Str$(mi.stats.icmpOutStats.dwParmProbs))
List1.AddString "[out] ParmProbs   : " & Item

'[out] SrcQuenchsを表示
Item = Trim$(Str$(mi.stats.icmpOutStats.dwSrcQuenchs))
List1.AddString "[out] SrcQuenchs  : " & Item

'[out] Redirectsを表示
Item = Trim$(Str$(mi.stats.icmpOutStats.dwRedirects))
List1.AddString "[out] SrcQuenchs  : " & Item

'[out] Echosを表示
Item = Trim$(Str$(mi.stats.icmpOutStats.dwEchos))
List1.AddString "[out] Echos       : " & Item

'[out] EchoRepsを表示
Item = Trim$(Str$(mi.stats.icmpOutStats.dwEchoReps))
List1.AddString "[out] EchoReps    : " & Item

'[out] Timestampsを表示
Item = Trim$(Str$(mi.stats.icmpOutStats.dwTimestamps))
List1.AddString "[out] Timestamps  : " & Item

'[out] TimestampRepsを表示
Item = Trim$(Str$(mi.stats.icmpOutStats.dwTimestampReps))
List1.AddString "[out] TimestampReps: " & Item

'[out] AddrMasksを表示
Item = Trim$(Str$(mi.stats.icmpOutStats.dwAddrMasks))
List1.AddString "[out] AddrMasks   : " & Item
```

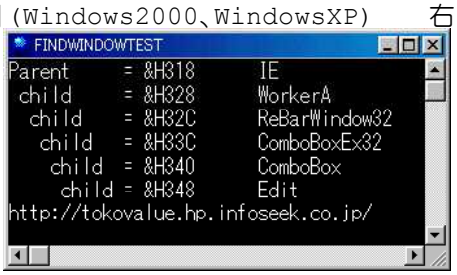
```
    '[out] AddrMaskRepsを表示
    Item = Trim$(Str$(mi.stats.icmpOutStats.dwAddrMaskReps))
    List1.AddString "[out] AddrMaskReps : " & Item
End Sub

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

IEのアドレス欄にある文字列の取得

IEのアドレスにある文字列を取得します。
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のハンドルを返す
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
GetWindowText ウインドウのタイトル文字列を取得
SendMessage ウィンドウにメッセージを送信

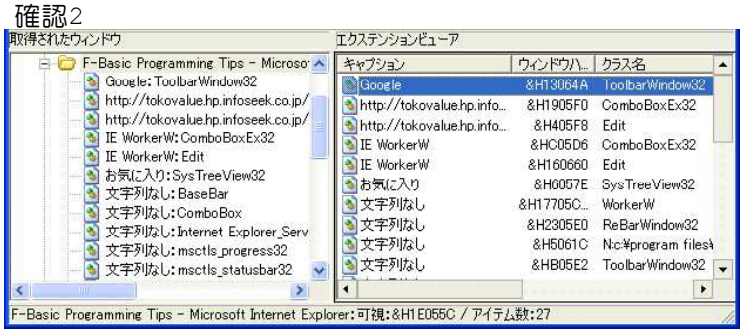
例では、IEのハンドルを取得しチャイルドのハンドルをたどって最終のアドレス欄(EDIT)のクラスにある文字列を取得しています。
Windows9x系とWindowsNT系ではクラス名が異なる個所があるためOSにより切り替えています。
左:最初のChildクラス名「WorkerW」(Windows2000、WindowsXP)　　右:「WorkerA」(Windows98)





確認1

確認2



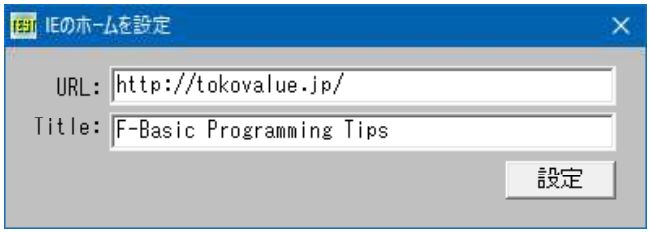

```
'=======================================
'= IEのアドレス欄にある文字列の取得
'=======================================
Type OSVERSIONINFO
    dwOSVersionInfoSize     As Long
    dwMajorVersion          As Long
    dwMinorVersion          As Long
    dwBuildNumber           As Long
    dwPlatformId            As Long
    szCSDVersion            As String * 128
End Type

#define VER_PLATFORM_WIN32_WINDOWS 1       'Windows9x
#define VER_PLATFORM_WIN32_NT 2            'WindowsNT、2000、XP

' 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)
```

```
' 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$,
ByVal lpWindowName$)

' 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&)

' 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 WM_GETTEXT &HD                          'コントロールのキャプション・テキストをバッファにコピー

Var strPlatForm As String
Var osvi As OSVERSIONINFO
Var ie As Long
Var child As Long
Var lpClass As String
Var txt As String
Var Ret As Long

osvi.dwOSVersionInfoSize = len(osvi)
Ret = Api_GetVersionEx(osvi)

If osvi.dwPlatformId = VER_PLATFORM_WIN32_NT Then
    lpClass = "WorkerW"
Else
    lpClass = "WorkerA"
End If

ie = Api_FindWindow("IEFrame", ByVal 0)
Print "Parent = &H" & Hex$(ie) & Chr$(9) & "IE"

child = Api_FindWindowEx(ie, 0, lpClass, ByVal 0)
Print " child = &H" & Hex$(child) & Chr$(9) & lpClass

child = Api_FindWindowEx(child, 0, "ReBarWindow32", ByVal 0)
Print " child = &H" & Hex$(child) & Chr$(9) & "ReBarWindow32"

child = Api_FindWindowEx(child, 0, "ComboBoxEx32", ByVal 0)
Print " child = &H" & Hex$(child) & Chr$(9) & "ComboBoxEx32"

child = Api_FindWindowEx(child, 0, "ComboBox", ByVal 0)
Print " child = &H" & Hex$(child) & Chr$(9) & "ComboBox"

child = Api_FindWindowEx(child, 0, "Edit", ByVal 0)
Print " child = &H" & Hex$(child) & Chr$(9) & "Edit"

txt = space$(250)
If child <> 0 Then
    Ret = Api_GetWindowText(child, txt, len(txt))
    Ret = Api_SendMessage(child, WM_GETTEXT, len(txt), txt)
    Print Left$(txt, InStr(txt, Chr$(0)) - 1)
Else
    Print "エラー"
End If

Stop
End
```

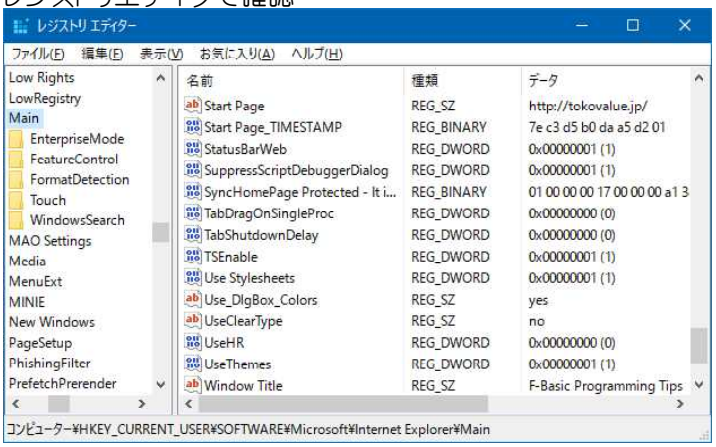
IEのホーム・タイトルを設定

インターネットエクスプローラの「ホーム」および「タイトル」を設定します。
RegCreateKey レジストリキーを作成
RegCloseKey レジストリのハンドルを解放
RegSetValueEx レジストリキーの値を設定

URLおよびタイトルを入力し、「設定」をクリックします。画面上の変化はありませんが、レジストリエディタで書き換わっていることを確認しています。
IEを再起動すると書き換わっているのが確認できます。



レジストリエディタで確認




```
'=======================================
'= IEのホームを設定
'=    (IEStartPage.bas)
'=======================================
#include "Windows.bi"

' レジストリキーを作成
Declare Function Api_RegCreateKey& Lib "advapi32" Alias "RegCreateKeyA" (ByVal hKey&,
ByVal lpSubKey$, phkResult&)

' レジストリのハンドルを解放
Declare Function Api_RegCloseKey& Lib "advapi32" Alias "RegCloseKey" (ByVal hKey As Long)

' 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&)

#define REG_SZ 1                                '文字列型
#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 ERROR_SUCCESS 0                         '正常終了の戻り値を示す

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

'=======================================
'=
'=======================================
Declare Sub IEStartURL(URL As String, Title As String)
Sub IEStartURL(URL As String, Title As String)
    Var hKey As Long
    Var RPath As String
    Var Ret As Long

    RPath = "Software¥Microsoft¥Internet Explorer¥Main"

    Ret = Api_RegCreateKey(HKEY_CURRENT_USER, RPath, hKey)

    If Ret = ERROR_SUCCESS Then
        Ret = Api_RegSetValueEx(hKey, "Start Page", 0, REG_SZ, URL, Len(URL))
        Ret = Api_RegCloseKey(hKey)
    End If

    Ret = Api_RegCreateKey(HKEY_CURRENT_USER, RPath, hKey)
```

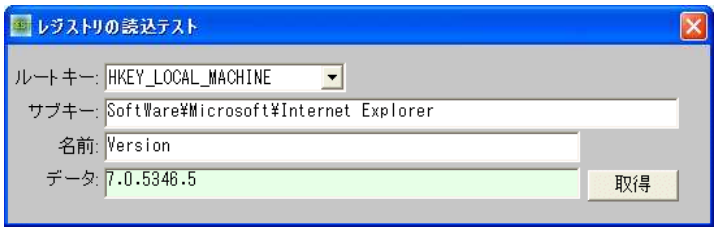
```
    If Ret = ERROR_SUCCESS Then
        Ret = Api_RegSetValueEx(hKey, "Window Title", 0, REG_SZ, Title, Len(Title))
        Ret = Api_RegCloseKey(hKey)
    End If
End Sub

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var URL As String
    Var Title As String

    URL = Edit(0).GetWindowText
    Title = Edit(1).GetWindowText
    IEStartURL URL, Title
End Sub

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

IEのバージョン取得

インターネットエクスプローラのバージョンを取得します。Windows Me, 2000以降か、IE5以降
Version5、Version6はこれで取得できたのですが••(IE7β版はこれでは取得できません)レジストリから取得した方が良いようです。
DllGetVersion Shell32.dllのバージョン取得

WinsowsXPでの

Windows98のIEバージョンでの例




```
'=======================================
'= IEのバージョン取得
'=    (DllGetVersion.bas)
'=======================================
#include "Windows.bi"

#define DLLVER_PLATFORM_WINDOWS &H1       'Windows 95
#define DLLVER_PLATFORM_NT &H2            'Windows NT

Type DllVersionInfo
   cbSize             As Long
   dwMajorVersion     As Long
```

```
   dwMinorVersion     As Long
   dwBuildNumber      As Long
   dwPlatformID       As Long
End Type

Declare Function Api_DllGetVersion& Lib "shlwapi" Alias "DllGetVersion" (dwVersion As
DllVersionInfo)

Var Shared Bitmap As Object
Var Shared Text1 As Object
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
BitmapObject Bitmap

'=======================================
'=
'=======================================
Declare Function GetIEVersionString() As String
Function GetIEVersionString() As String
    Var Ret As Long
    Var DVI As DllVersionInfo

    DVI.cbSize = Len(DVI)
    Ret = Api_DllGetVersion(DVI)

    GetIEVersionString = "Version " & Trim$(Str$(DVI.dwMajorVersion)) & "." & Trim$(Str$
(DVI.dwMinorVersion)) & "." & Trim$(Str$(DVI.dwBuildNumber))
End Function

'=======================================
'=
'=======================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Bitmap.LoadFile "IELogo.bmp"
    DrawBitmap Bitmap,10,10
    Bitmap.DeleteObject

    Text1.SetWindowText GetIEVersionString
End Sub

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

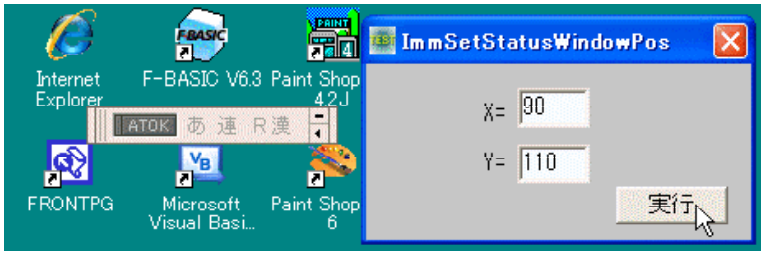
IE(複数起動)の終了

IEが複数起動されている場合一度に終了させています。
GetWindow ウィンドウハンドルを取得
GetClassName クラス名を取得
SendMessage ウィンドウにメッセージを送信

```
'=====================================
'= IE(複数)を終了させる
'=    (IEClose.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 WM_SYSCOMMAND &H112                 'システムメニューが操作された
#define SC_CLOSE &HF060                     '閉じる
#define GW_HWNDFIRST 0                      '最前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

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

    '最前面のウィンドウハンドルを取得
    hWnd = Api_GetWindow(GethWnd, GW_HWNDFIRST)

    '見つかったなら、クラス名が"IEFrame"出あれば「終了」を繰り返す
    Do While hWnd <> 0
        ClassName = String$(256, " ")

        'クラス名を取得
        Ret = Api_GetClassName(hWnd, ClassName, 255)
        ClassName = Left$(ClassName, InStr(ClassName, Chr$(0)) - 1)

        If ClassName = "IEFrame" Then
            Ret = Api_SendMessage(hWnd, WM_SYSCOMMAND, SC_CLOSE, 0)
        End If

        hWnd = Api_GetWindow(hWnd, GW_HWNDNEXT)
    Loop
End Sub

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

IMEステータスウィンドウの位置を設定

IMEステータスウィンドウ(ATOKでは、ATOKパレット)の位置を設定します。
ImmGetContext 指定したウィンドウの入力コンテキストのハンドルを取得
ImmSetStatusWindowPos ステータスウィンドウの位置を設定



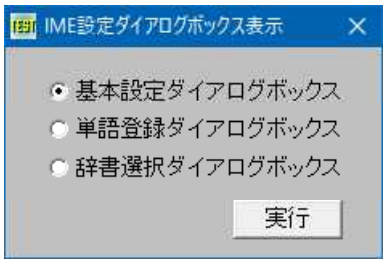

```
'=====================================
'= IMEステータスウィンドウの位置を設定
'=    (ImmSetStatusWindowPos.bas)
'=====================================
#include "Windows.bi"

Type POINTAPI
    X   As Long
    Y   As Long
End Type

' 指定したウィンドウの入力コンテキストのハンドルを取得
Declare Function Api_ImmGetContext& Lib "imm32" Alias "ImmGetContext"(ByVal hWnd&)

' ステータスウィンドウの位置を設定
Declare Function Api_ImmSetStatusWindowPos& Lib "imm32" Alias "ImmSetStatusWindowPos"
(ByVal himc&, lpPoint As POINTAPI)

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

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

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hWnd As Long
    Var pa As POINTAPI
    Var Ret As Long

    hWnd = Api_ImmGetContext(GethWnd)
    pa.X = Val(Edit(0).GetWindowText)
    pa.Y = Val(Edit(1).GetWindowText)

    Ret = Api_ImmSetStatusWindowPos(hWnd, pa)
    SetFocus
End Sub

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

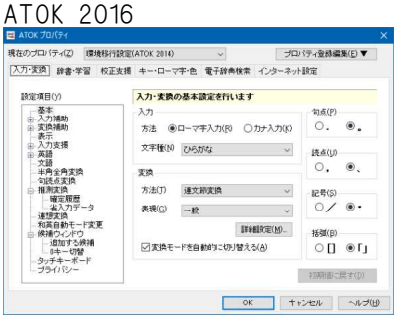
IME設定ダイアログボックスの表示

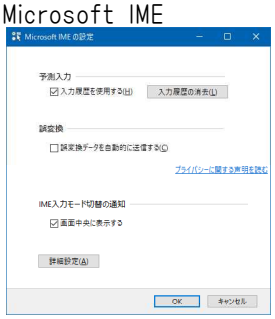
IME設定ダイアログボックスを表示します。
GetKeyboardLayout キーボードレイアウトのハンドルを取得
ImmConfigureIME IMEの環境設定ダイアログを表示する関数

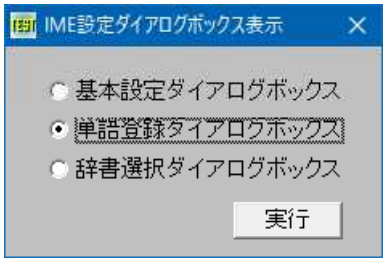
ATOK 2016

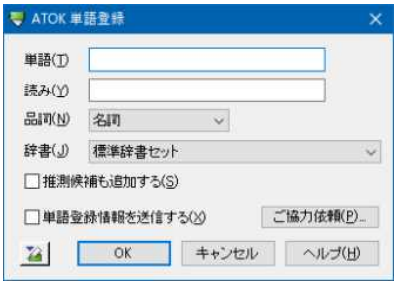
Microsoft IME

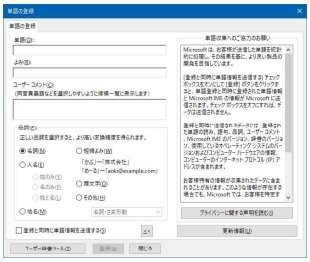


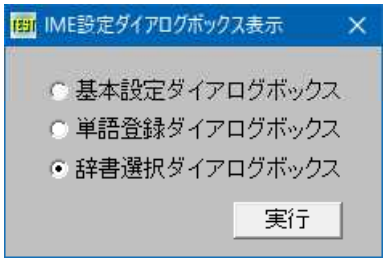


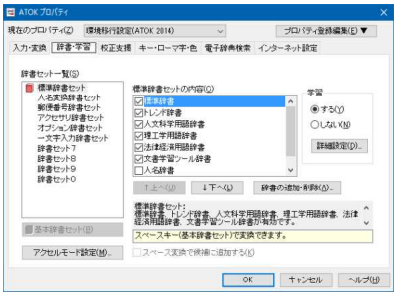


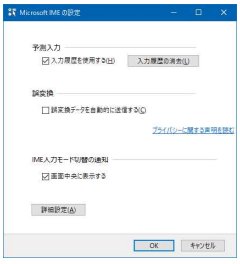












基本設定ダイアログボックスが表示されるので、「辞書・学習」タブをクリックします。

```
'=======================================
'= IMEの設定ダイアログボックスの表示
'=    (ImmConfigureIME.bas)
'=======================================
#include "Windows.bi"

' キーボードレイアウトのハンドルを取得
Declare Function Api_GetKeyboardLayout& Lib "user32" Alias "GetKeyboardLayout" (ByVal
dwLayout&)

' IMEの環境設定ダイアログを表示する関数
Declare Function Api_ImmConfigureIME& Lib "imm32" Alias "ImmConfigureIMEA" (ByVal hKL&,
ByVal hWnd&, ByVal dwMode&, lpdata As Any)

#define IME_CONFIG_GENERAL 1                '基本設定ダイアログボックスを表示
#define IME_CONFIG_REGISTERWORD 2           '単語登録ダイアログボックスを表示
#define IME_CONFIG_SELECTDICTIONARY 3       '辞書選択ダイアログボックスを表示

' 単語登録の読みと単語を定義する構造体
Type tagREGISTERWORD
    lpReading As String * 255
    lpWord    As String * 255
End Type

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

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
```

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

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var KeyHandle As Long
    Var DialogBoxType As Long
    Var RegWord As tagREGISTERWORD
    Var Ret As Long

    'カレントスレッドのキーボードレイアウトのハンドルを取得
    KeyHandle = Api_GetKeyboardLayout(0)

    '表示するIMEの環境設定ダイアログのタイプを設定
    Select Case Index
        Case 1
            DialogBoxType = IME_CONFIG_GENERAL
        Case 2
            DialogBoxType = IME_CONFIG_REGISTERWORD
        Case 3
            DialogBoxType = IME_CONFIG_SELECTDICTIONARY
    End Select

    'IME設定ダイアログを表示
    Ret = Api_ImmConfigureIME(KeyHandle, GethWnd, DialogBoxType, RegWord)
End Sub

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

IMEのWindowsバージョンを取得

GetKeyboardLayout キーボードレイアウトのハンドルを取得
ImmGetProperty 指定された入力ロケールに関連付けられているIMEのプロパティや機能に関する情報を取得


```
'=====================================
'= IMEのWindowsバージョンを取得
'=    (ImmGetProperty.bas)
'=====================================
#include "Windows.bi"

' キーボードレイアウトのハンドルを取得
Declare Function Api_GetKeyboardLayout& Lib "user32" Alias "GetKeyboardLayout" (ByVal
dwLayout&)
```

```
' 指定された入力ロケールに関連付けられている IME のプロパティや機能に関する情報を取得
Declare Function Api_ImmGetProperty& Lib "imm32" Alias "ImmGetProperty" (ByVal hkl&,
ByVal dw&)

#define IGP_CONVERSION &H8                  '変換関連の機能
#define IGP_GETIMEVERSION -&H4              '指定したIMEに対応するシステムバージョン番号
#define IGP_PROPERTY &H4                    'プロパティ情報
#define IGP_SELECT &H18                     '選択継承機能
#define IGP_SENTENCE &HC                    '変換モードの機能
#define IGP_SETCOMPSTR &H14                 '変換文字列関連の機能
#define IGP_UI &H10                         'ユーザーインターフェイス関連の機能
#define IMEVER_0310 &H3000A                 'Windows3.1用に作成されたIME
#define IMEVER_0400 &H40000                 'Windows95用に作成されたIME

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 hKeyLayout As Long
    Var IMEWinVersion As Long

    'カレントスレッドのキーボードレイアウトのハンドル取得
    hKeyLayout = Api_GetKeyboardLayout(0)

    'IMEのWindowsバージョン取得
    IMEWinVersion = Api_ImmGetProperty(hKeyLayout, IGP_GETIMEVERSION)

    'IMEのWindowsバージョン表示
    Select Case IMEWinVersion
        Case IMEVER_0310
            Text2.SetWindowText "Windows 3.1用に作成されました。"
        Case IMEVER_0400
            Text2.SetWindowText "Windows 95用に作成されました。"
        Case Else
            Text2.SetWindowText "取得できません。"
    End Select
End Sub

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

IMEの種類を取得(Windows10不可)

IMEの種類を取得します。
GetKeyboardLayout キーボードレイアウトのハンドルを取得
ImmGetDescription IMEの種類を取得

Windows10 では取得できませんでした。

```
'=====================================
'= IMEの種類を取得
'=    (ImmGetDescription.bas)
'=====================================
#include "Windows.bi"

' キーボードレイアウトのハンドルを取得
Declare Function Api_GetKeyboardLayout& Lib "user32" Alias "GetKeyboardLayout" (ByVal
dwLayout&)

' IMEの種類を取得
Declare Function Api_ImmGetDescription& Lib "imm32" Alias "ImmGetDescriptionA" (ByVal
hKL&, ByVal lpszDescription$, ByVal uBufLen&)

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 hKeyLayout As Long
    Var IMEDescrpt As String * 128
    Var Ret As Long

    'キーボードレイアウトのハンドルを取得
    hKeyLayout = Api_GetKeyboardLayout(0)

    'IMEの種類を取得
    Ret = Api_ImmGetDescription(hKeyLayout, IMEDescrpt, Len(IMEDescrpt))

    'IMEの種類を表示
    Text2.SetWindowText Left$(IMEDescrpt, InStr(IMEDescrpt, Chr$(0)) - 1)
End Sub

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

IME初期方式を取得(Windows10不可)

IME初期方式を取得します。
ImmGetContext ウィンドウに関連付けされた入力コンテキストを取得
ImmReleaseContext ウィンドウに関連付けされた入力コンテキストを開放
ImmGetConversionStatus IME初期方式を取得する関数


```
'=====================================
'= IME初期方式を取得
'=    (ImmGetConversionStatus.bas)
'=====================================
#include "Windows.bi"

' ウィンドウに関連付けされた入力コンテキストを取得
Declare Function Api_ImmGetContext& Lib "imm32" Alias "ImmGetContext" (ByVal hWnd&)

' ウィンドウに関連付けされた入力コンテキストを開放
Declare Function Api_ImmReleaseContext& Lib "imm32" Alias "ImmReleaseContext" (ByVal
hWnd&, ByVal hIMC&)

' IME初期方式を取得
Declare Function Api_ImmGetConversionStatus& Lib "imm32" Alias "ImmGetConversionStatus"
(ByVal hIMC&, fdwConversion&, fdwSentence&)

#define IME_CMODE_ALPHANUMERIC &H0          '英数字モード
#define IME_CMODE_CHARCODE &H20             '文字コード入力
#define IME_CMODE_CHINESE &H1               'NATIVEモード(設定しない場合ALPHANUMERICモード)
#define IME_CMODE_EUDC &H200                'EUDC変換モード
#define IME_CMODE_FULLSHAPE &H8             '全角モード(設定しないとき､半角モード)
#define IME_CMODE_HANGEUL &H1               'NATIVEモード(設定しない場合ALPHANUMERICモード)
#define IME_CMODE_HANJACONVERT &H40         'HANJA変換モード
#define IME_CMODE_JAPANESE &H1              'NATIVEモード(設定しない場合ALPHANUMERICモード)
#define IME_CMODE_KATAKANA &H2              'カタカナモード(設定しない場合ひらがなモード)
#define IME_CMODE_LANGUAGE &H3              '
#define IME_CMODE_NATIVE &H1                'NATIVEモード(設定しない場合ALPHANUMERICモード)
#define IME_CMODE_NOCONVERSION &H100        '変換しない(設定しないとき､変換する)
#define IME_CMODE_ROMAN &H10                'ローマ字変換モード
#define IME_CMODE_SOFTKBD &H80              'ソフトキーボードモード
#define IME_CMODE_SYMBOL &H400              'シンボルモード

Var Shared Text(3) As Object
Var Shared Edit1 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

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

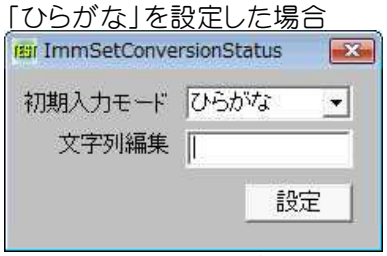
```
'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hWnd  As Long
    Var hIMC  As Long
    Var IMEConversion As Long
    Var IMESentence  As Long
    Var Ret  As Long

    'ウィンドウに関連付けされた入力ハンドル･コンテキストを取得
    hWnd = Edit1.GethWnd
    hIMC = Api_ImmGetContext(hWnd)

    '入力コンテキストを取得できたときは
    If hIMC <> 0 Then

        'IME初期方式の設定を取得
        Ret = Api_ImmGetConversionStatus(hIMC, IMEConversion, IMESentence)

        'IME初期入力方式の設定を表示
        Select Case IMEConversion And IME_CMODE_ROMAN
            Case IME_CMODE_ROMAN
                Text(1).SetWindowText "ローマ字入力"
            Case Else
                Text(1).SetWindowText "かな入力"
        End Select

        'IME初期入力モードの設定を表示
        Select Case IMEConversion And (IME_CMODE_JAPANESE Or IME_CMODE_KATAKANA Or
IME_CMODE_LANGUAGE Or IME_CMODE_FULLSHAPE)
            Case IME_CMODE_FULLSHAPE Or IME_CMODE_JAPANESE
                Text(3).SetWindowText "ひらがな"
            Case IME_CMODE_FULLSHAPE Or IME_CMODE_LANGUAGE
                Text(3).SetWindowText "全角カタカナ"
            Case IME_CMODE_LANGUAGE
                Text(3).SetWindowText "半角カタカナ"
            Case IME_CMODE_FULLSHAPE
                Text(3).SetWindowText "全角英数"
            Case IME_CMODE_ALPHANUMERIC
                Text(3).SetWindowText "半角英数"
            Case Else
                Text(3).SetWindowText "不明"
        End Select

        '入力コンテキストを開放
        Ret = Api_ImmReleaseContext(hWnd, hIMC)
    End If
End Sub

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

IMEの初期方式を設定(Windows10不可)

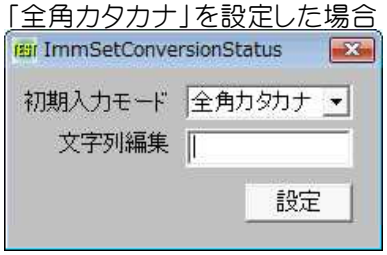
IMEの初期方式を設定します。
ImmGetContext ウィンドウに関連付けされた入力コンテキストを取得
ImmReleaseContext ウィンドウに関連付けされた入力コンテキストを開放
ImmGetOpenStatus IMEのオープン状態を取得
ImmSetOpenStatus IMEのオープン状態を設定

ImmSetConversionStatus 現在の変換状態を設定

「ひらがな」を設定した場合



「全角カタカナ」を設定した場合




```
'=====================================
'= IMEの初期方式を設定
'=    (ImmSetConversionStatus.bas)
'=====================================
#include "Windows.bi"

' ウィンドウに関連付けされた入力コンテキストを取得
Declare Function Api_ImmGetContext& Lib "imm32" Alias "ImmGetContext" (ByVal hWnd&)

' ウィンドウに関連付けされた入力コンテキストを開放
Declare Function Api_ImmReleaseContext& Lib "imm32" Alias "ImmReleaseContext" (ByVal
hWnd&, ByVal hIMC&)

' IMEのオープン状態を取得
Declare Function Api_ImmGetOpenStatus& Lib "imm32" Alias "ImmGetOpenStatus" (ByVal
hIMC&)

' IMEのオープン状態を設定
Declare Function Api_ImmSetOpenStatus& Lib "imm32" Alias "ImmSetOpenStatus" (ByVal
hIMC&, ByVal fOpen&)

' 現在の変換状態を設定
Declare Function Api_ImmSetConversionStatus& Lib "imm32" Alias "ImmSetConversionStatus"
(ByVal hIMC&, ByVal fdwConversion&, ByVal fdwSentence&)

' 入力モードを示す定数の宣言
#define IME_CMODE_CHARCODE &H20             '文字コード入力
#define IME_CMODE_CHINESE &H1               'NATIVEモード(設定しない場合ALPHANUMERICモード)
#define IME_CMODE_EUDC &H200                'EUDC変換モード
#define IME_CMODE_FULLSHAPE &H8             '全角モード(設定しないとき､半角モード)
#define IME_CMODE_HANGEUL &H1               'NATIVEモード(設定しない場合ALPHANUMERICモード)
#define IME_CMODE_HANJACONVERT &H40         'HANJA変換モード
#define IME_CMODE_JAPANESE &H1              'NATIVEモード(設定しない場合ALPHANUMERICモード)
#define IME_CMODE_KATAKANA &H2              'カタカナモード(設定しない場合ひらがなモード)
#define IME_CMODE_LANGUAGE &H3              '
#define IME_CMODE_NATIVE &H1                'NATIVEモード(設定しない場合ALPHANUMERICモード)
#define IME_CMODE_NOCONVERSION &H100        '変換しない(設定しないとき､変換する)
#define IME_CMODE_ROMAN &H10                'ローマ字変換モード
#define IME_CMODE_SOFTKBD &H80              'ソフトキーボードモード
#define IME_CMODE_SYMBOL &H400              'シンボルモード
#define IME_CONFIG_GENERAL 1                '
```

```
#define IME_CONFIG_REGISTERWORD 2          '
#define IME_CONFIG_SELECTDICTIONARY 3

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hIMC As Long
    Var ConversionStatus As Long
    Var SentenceStatus As Long
    Var Ret As Long

    'ウィンドウに関連付けされた入力コンテキストを取得
    hIMC = Api_ImmGetContext(GethWnd)

    '入力コンテキストを取得できたときは
    If hIMC <> 0 Then

        'IMEのオープン状態でないときは
        If Api_ImmGetOpenStatus(hIMC) <> Not False Then

            'IMEをオープン
            Ret = Api_ImmSetOpenStatus(hIMC, Not False)

            '初期入力モードを指定
            ConversionStatus = iStatus(Combo1.GetCursel)

            '初期変換モードを指定
            SentenceStatus = IME_SMODE_PHRASEPREDICT

            'IMEの初期方式を設定
            Ret = Api_ImmSetConversionStatus(hIMC, ConversionStatus, SentenceStatus)

            'IMEをクローズ
            Ret = Api_ImmSetOpenStatus(hWnd, False)
        End If

        '入力コンテキストを開放
        Ret = Api_ImmReleaseContext(GethWnd, hIMC)
    End If

    SetImeMode 1
    Edit1.SetFocus
End Sub

'=====================================
'=
'=====================================
Declare Sub Edit1_SetFocus edecl ()
Sub Edit1_SetFocus()
    Edit1.SetSelText 0, 0
End Sub

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

IMEのファイル名を取得(Windows10不可)

IMEのファイル名を取得します。
GetKeyboardLayout キーボードレイアウトのハンドルを取得

ImmGetIMEFileName IMEのファイル名を取得する関数の宣言

例:Valuestar:WindowsXP

Flora:Windows2000

Flora:Windows98


```
'=====================================
'= IMEファイル名を取得
'=    (ImmGetIMEFileName.bas)
'=====================================
#include "Windows.bi"

' キーボードレイアウトのハンドルを取得
Declare Function Api_GetKeyboardLayout& Lib "user32" Alias "GetKeyboardLayout" (ByVal
dwLayout&)

' IMEのファイル名を取得する関数の宣言
Declare Function Api_ImmGetIMEFileName& Lib "imm32" Alias "ImmGetIMEFileNameA" (ByVal
hKL&, ByVal lpszFileName$, ByVal uBufLen&)

Var Shared Text(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
Next
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var KeybdhWnd As Long
    Var ImeFileName As String * 128
    Var Ret As Long

    KeybdhWnd = Api_GetKeyBoardLayout(0)
    Ret = Api_ImmGetIMEFileName(KeybdhWnd, ImeFileName, Len(ImeFileName))

    Text(1).SetWindowText Left$(ImeFileName, InStr(ImeFileName, Chr$(0)) - 1)
End Sub

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

iniファイルからキーやセクションを削除

iniファイルからキーやセクションを削除します。
WritePrivateProfileString iniファイルからキーやセクションを削除

図のようなiniファイル(C:¥test¥test.ini)があるとします。

[Color]というセクションを指定して削除ボタンをクリックします。[Color]セクションが削除されているのが確認できます。

[URL]というセクションを指定して削除ボタンをクリックします。[URL]セクションが削除されているのが確認できます。

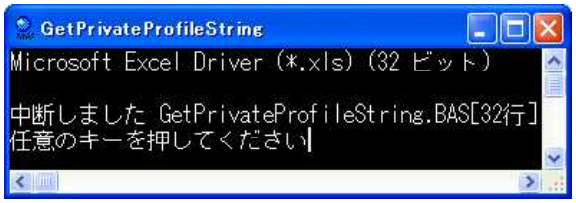

```
'=======================================
'= iniファイルからキーやセクションを削除
'=    (WritePrivateProfileString.bas)
'=======================================
#include "Windows.bi"

' iniファイルからキーやセクションを削除
Declare Function Api_WritePrivateProfileString& Lib "kernel32" Alias
"WritePrivateProfileStringA" (ByVal Section$, ByVal NoKey&, ByVal NoSetting&, ByVal
FileName$)

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

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

Var Shared SectionName As String
Var Shared IniFile As String

'=======================================
'=
'=======================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    IniFile = "c:¥test¥test.ini"
    SectionName = "Color"

    Text(2).SetWindowText IniFile
    Edit1.SetWindowText SectionName
End Sub

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

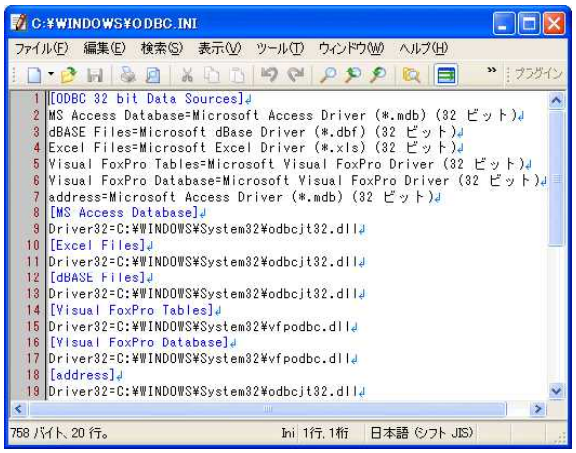
```
Sub Button1_on()
    Var Ret As Long

    SectionName = Edit1.GetwindowText
    Ret = Api_WritePrivateProfileString(SectionName, 0, 0, IniFile)
End Sub

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

iniファイルから文字列を取得

GetPrivateProfileString iniファイルから文字列を取得
WritePrivateProfileString iniファイルに文字列を書き込む

例では、C:¥Windows¥odbc.ini(下図参照)から取得しています。
AppName = "ODBC 32 bit Data Sources"
KeyName = "Excel Files"
FileName = "c:¥windows¥odbc.ini" で取得しています。






```
'=======================================
'= iniファイルから文字列を取得
'=    (GetPrivateProfileString.bas)
'=======================================

' iniファイルから文字列を取得
Declare Function Api_GetPrivateProfileString& Lib "kernel32" Alias
"GetPrivateProfileStringA" (ByVal lpApplicationName$, ByVal lpKeyName As Any, ByVal
lpDefault$, ByVal lpBuffurnedString$, ByVal nSize&, ByVal lpFileName$)

' iniファイルに文字列を書き込む
Declare Function Api_WritePrivateProfileString& Lib "kernel32" Alias
"WritePrivateProfileStringA" (ByVal lpApplicationName$, ByVal lpKeyName As Any, ByVal
lpString As Any, ByVal lpFileName$)

Var AppName As String
Var KeyName As String
```

```
Var Default As String
Var BuffString As String
Var FileName As String
Var Leng As Long
Var Ret As Long

AppName = "ODBC 32 bit Data Sources"
KeyName = "Excel Files"
FileName = "c:¥windows¥odbc.ini"

BuffString = String$(256, Chr$(0))
Leng = Len(BuffString)

Ret = Api_GetPrivateProfileString(AppName, KeyName, Default, BuffString, Leng, FileName)

If Ret <> 0 Then BuffString = Left$(BuffString, Ret)

Print BuffString

Stop
End

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

書き込む場合は、

```
Ret = Api_WritePrivateProfileString(AppName, KeyName, WorkString, FileName)
```

IPアドレスからホスト名を取得

IPアドレスからホスト名を取得します。
WSAStartup WinSockを初期化
WSACleanup WinSock のリソースを解放
inet_addr 文字列を32ビットのバイナリ値に変換
gethostbyaddr　IPアドレスをホスト名に変換する関数
RtlMoveMemory メモリブロックのコピーを行うための関数
lstrlen 文字列の長さを調べる

アドレスは、E-Mailのプロパティ → 詳細 → インターネットヘッダー のIPアドレスを入力し結果の確認をしています。

例1　IPアドレスを入力しない状態

例2

例3

例4　適当な数字を入れてみた


```
'=======================================
'= IPアドレスからホスト名を取得
'=    (gethostbyaddr.bas)
'=======================================
#include "Windows.bi"

#define WSADescription_Len 256
#define WSASYS_Status_Len 128
#define WS_VERSION_REQD &H101
#define IP_SUCCESS 0
#define SOCKET_ERROR  -1
#define AF_INET  2
```

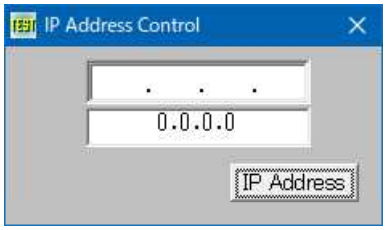
```
Type WSADATA
    wVersion          As Integer
    wHighVersion      As Integer
    szDescription(WSADescription_Len)  As Byte
    szSystemStatus(WSASYS_Status_Len)  As Byte
    imaxsockets       As Integer
    imaxudp           As Integer
    lpszvEnderinfo    As Long
End Type

' WinSockを初期化
Declare Function Api_WSAStartup& Lib "wsock32" Alias "WSAStartup" (ByVal VersionReq&,
WSADataReturn As WSADATA)

' WinSock のリソースを解放
Declare Function Api_WSACleanup& Lib "wsock32" Alias "WSACleanup" ()

' 文字列を32ビットのバイナリ値に変換
Declare Function Api_inet_addr& Lib "wsock32" Alias "inet_addr" (ByVal s$)

' IP アドレスをホスト名に変換する関数
Declare Function Api_gethostbyaddr& Lib "wsock32" Alias "gethostbyaddr" (haddr&, ByVal
hnlen&, ByVal addrtype&)

' メモリブロックのコピーを行うための関数
Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (xDest As Any, xSource As Any,
ByVal nbytes&)

' 文字列の長さを調べる
Declare Function Api_lstrlen& Lib "kernel32" Alias "lstrlenA" (lpString As Any)

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

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 Function SocketsInitialize () As Integer
Function SocketsInitialize() As Integer
    Var WSAD As WSADATA

    SocketsInitialize = Api_WSAStartup(WS_VERSION_REQD, WSAD) = IP_SUCCESS
End Function

'=====================================
'=
'=====================================
Declare Sub SocketsCleanup ()
Sub SocketsCleanup()
    If Api_WSACleanup() <> 0 Then
        A% = MessageBox("", "CleanUpエラー！", 0, 2)
    End If
End Sub

'=====================================
'=
'=====================================
Declare Function GetHostNameFromIP(sAddress As String) As String
Function GetHostNameFromIP(sAddress As String) As String
    Var ptrHosent As Long
    Var hAddress As Long
    Var nbytes As Long
```

```
    Var Ret As Long

    If SocketsInitialize() Then
        hAddress = Api_inet_addr(sAddress)

        If hAddress <> SOCKET_ERROR Then
            ptrHosent = Api_gethostbyaddr(hAddress, 4, AF_INET)

            If ptrHosent <> 0 Then
                CopyMemory ptrHosent, ByVal ptrHosent, 4
                nbytes = Api_lstrLen(ByVal ptrHosent)

                If nbytes > 0 Then
                    sAddress = Space$(nbytes)
                    CopyMemory sAddress, ByVal ptrHosent, nbytes
                    GetHostNameFromIP = sAddress
                End If
            Else
                A% = MessageBox("", "取得に失敗しました！", 0, 2)
            End If

            SocketsCleanup
        Else
            A% = MessageBox("", "無効のIPです！", 0, 2)
        End If
    Else
        A% = MessageBox("", "ソケットの初期化失敗！", 0, 2)
    End If
End Function

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hName As String

    Text(2).SetWindowText ""
    hName = GetHostNameFromIP(GetDlgItemText("Edit1"))
    Text(2).SetWindowText hName
End Sub

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

IPアドレスコントロール

ウィンドウ(コントロール)をコードで作成します。
InitCommonControlsEx 特定のコモンコントロールを作成
CreateWindowEx ウィンドウ(コントロール)を作成
GetWindowText ウィンドウのタイトル文字列を取得
SetWindowText ウィンドウのタイトルを変更
SetFocus ウィンドウにフォーカスを設定
GetClientRect ウィンドウのクライアント領域の座標を取得

PictureBoxをIPアドレスコントロールとして作成します。



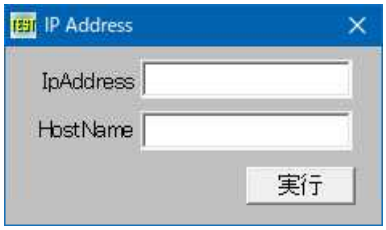
PictureBoxの代わりにEditBox、TextBoxでもOKです。その場合は、
IpInp = Api_CreateWindowEx(0, WC_IPADDRESS, ByVal 0, WS_CHILD Or WS_VISIBLE Or WS_OVERLAPPED, 0, 0, rct.Right, rct.Bottom, Text1.GethWnd, 0, GethInst, ByVal 0)
のようにコントロールのハンドルを指定します。

```
'=====================================
'= IPアドレスコントロール
'=    (IpAddress4.bas)
'=====================================
#include "Windows.bi"

Type INITCOMMONCONTROLSEX
    dwSize      As Long
    dwICC       As Long
End Type

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

#define WC_IPADDRESS "SysIPAddress32"       'IPアドレスコントロール
#define ICC_INTERNET_CLASSES &H800          'IPアドレスコントロール(Version 4.71 以降)
#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_OVERLAPPED &H0                   'オーバーラップウィンドウを作成する
#define WS_VISIBLE &H10000000               '可視状態のウィンドウを作成する
#define vbCrLf (Chr$(13) & Chr$(10))        'キャリッジリターンとラインフィード(¥r¥n)

' 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 Function Api_InitCommonControlsEx& Lib "comctl32" Alias "InitCommonControlsEx"
(lpInitCtrls As INITCOMMONCONTROLSEX)

' 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&)

' ウィンドウのタイトル文字列を取得
Declare Function Api_GetWindowText& Lib "user32" Alias "GetWindowTextA" (ByVal hWnd&,
ByVal lpString$, ByVal cch&)

' 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$)

' ウィンドウにフォーカスを設定
Declare Function Api_SetFocus& Lib "user32" Alias "SetFocus" (ByVal hWnd&)

' 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 IpInp As Long

Var Shared Picture1 As Object
Var Shared Text1 As Object
Var Shared Button1 As Object

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

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

'=====================================
'=
'=====================================
Declare Sub Picture1_MouseMove edecl ()
Sub Picture1_MouseMove()
    Var Ret As Long

    Ret = Api_SetFocus(IpInp)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var rct As RECT
    Var icc As INITCOMMONCONTROLSEX
    Var Ret As Long

    Ret = Api_GetClientRect(Picture1.GethWnd, rct)
    icc.dwSize = Len(icc)
    icc.dwICC = ICC_INTERNET_CLASSES

    Ret = Api_InitCommonControlsEx(icc)
    IpInp = Api_CreateWindowEx(0, WC_IPADDRESS, ByVal 0, WS_CHILD Or WS_VISIBLE Or
WS_OVERLAPPED, 0, 0, rct.Right, rct.Bottom, Picture1.GethWnd, 0, GethInst, ByVal 0)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var IPAddr As String * 15
    Var Ret As Long

    Ret = Api_GetWindowText(IpInp, IPAddr, 15)
    Text1.SetWindowText Left$(IPAddr, InStr(1, IPAddr, Chr$(0)) - 1)
End Sub

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

IPアドレス取得(Ⅰ)

IPアドレスとホスト名を取得します。
WSAStartup WinSockの初期化
WSACleanup WinSockのリソースを解放
WSAGetLastError WinSock で発生したエラー取得
gethostname ローカルマシンのホスト名を取得
gethostbyname ホスト名からホストに関する情報(HOSTENT構造体に対するポインタ)を得る



確認

■Xpの場合

コマンドプロンプト → ipconfig/release

コマンドプロンプト → ipconfig/renew

■Win9xの場合

「スタートメニュー」→「ファイル名を指定して実行」→「**winipcfg**」と入力「OK」→「IP設定」画面で確認

```
'=====================================
'= IPアドレス取得
'=    (IpAddress2.bas)
'=====================================
#include "Windows.bi"

#define MIN_SOCKETS_REQD 1
#define WS_VERSION_REQD &H101
#define WS_VERSION_MAJOR (WS_VERSION_REQD ¥ &H100 and &HFF)
#define WS_VERSION_MINOR (WS_VERSION_REQD and &HFF)

#define SOCKET_ERROR -1
#define WSADESCRIPTION_LEN 257
#define WSASYS_STATUS_LEN 129
#define MAX_WSADescription 256
#define MAX_WSASYSStatus 128

Type WSAData
    wVersion          As Integer
    wHighVersion      As Integer
    szDescription(MAX_WSADescription) As Byte
    szSystemStatus(MAX_WSASYSStatus)  As Byte
    wMaxSockets       As Integer
    wMaxUDPDG         As Integer
    dwVEndorInfo      As Long
End Type

Type HOSTENT
    hName      As Long
    hAliases   As Long
    hAddrType  As Integer
    hLen       As Integer
    hAddrList  As Long
End Type

'WinSockの初期化
Declare Function Api_WSAStartup& Lib "WSOCK32" Alias "WSAStartup" (ByVal
wVersionRequired&, lpWSADATA As WSAData)

'WinSockのリソースを解放
Declare Function Api_WSACleanup& Lib "WSOCK32" Alias "WSACleanup" ()

'WinSock で発生したエラー取得
Declare Function Api_WSAGetLastError& Lib "WSOCK32" Alias "WSAGetLastError" ()
```

```
'ローカルマシンのホスト名を取得
Declare Function Api_gethostname& Lib "WSOCK32" Alias "gethostname" (ByVal szHost$,
ByVal dwHostLen&)

'ホスト名からホストに関する情報( hostent 構造体に対するポインタ)を得る
Declare Function Api_gethostbyname& Lib "WSOCK32" Alias "gethostbyname" (ByVal szHost$)

' 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" (hpvDest As Any, ByVal
hpvSource&, ByVal cbCopy&)

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 sHostName As String * 256

'=====================================
'=
'=====================================
Declare Function HiByte(ByVal wParam As Integer)
Function HiByte(ByVal wParam As Integer)
    HiByte = wParam ¥ &H100 And &HFF
End Function

'=====================================
'=
'=====================================
Declare Function LoByte(ByVal wParam As Integer)
Function LoByte(ByVal wParam As Integer)
    LoByte = wParam And &HFF
End Function

'=====================================
'=
'=====================================
Declare Sub SocketsCleanup ()
Sub SocketsCleanup()
    If Api_WSACleanup() <> ERROR_SUCCESS Then
        A% = MessageBox("", "ソケットエラーをクリアしました！", 0, 2)
    End If
End Sub

'=====================================
'=
'=====================================
Declare Function SocketsInitialize() As Integer
Function SocketsInitialize() As Integer
    Var WSAD As WSAData
    Var sLoByte As String
    Var sHiByte As String

    If Api_WSAStartup(WS_VERSION_REQD, WSAD) <> ERROR_SUCCESS Then
        A% = MessageBox("", "32ビットWindowsソケットが反応しません！。", 0, 2)
        SocketsInitialize = False
        Exit Function
    End If

    If WSAD.wMaxSockets < MIN_SOCKETS_REQD Then
        A% = MessageBOx("", "このアプリケーションは、最小のサポートされたソケットを必要とします", 0, 2)
        SocketsInitialize = False
        Exit Function
    End If

    If LoByte(WSAD.wVersion) < WS_VERSION_MAJOR Or (LoByte(WSAD.wVersion) =
WS_VERSION_MAJOR and HiByte(WSAD.wVersion) < WS_VERSION_MINOR) Then
```

```
        sHiByte = Str$(HiByte(WSAD.wVersion))
        sLoByte = Str$(LoByte(WSAD.wVersion))
        A% = MessageBox("", "ソケットバージョン " & sLoByte & "." & sHiByte & " は、サポートされていません！", 0, 2)
        SocketsInitialize = False
        Exit Function
    End If

    SocketsInitialize = True
End Function

'=====================================
'=
'=====================================
Declare Function GetIPAddress() As String
function GetIPAddress() As String
    Var lpHost As Long
    Var HOST As HOSTENT
    Var dwIPAddr As Long
    Var i As Integer
    Var sIPAddr As String

    If Not SocketsInitialize Then
        GetIPAddress = ""
        Exit Function
    End If

    If Api_gethostname(sHostName, 256) = SOCKET_ERROR Then
        GetIPAddress = ""
        A% = MessageBox("", "ホスト名を取得できません！", 0, 2)
        SocketsCleanup
        Exit Function
    End If

    sHostName = Trim$(sHostName)
    lpHost = Api_gethostbyname(sHostName)
    If lpHost = 0 Then
        GetIPAddress = ""
        A% = MessageBox("", "Windowsソケットは反応していません。ホスト名を取得できません！", 0, 2)
        SocketsCleanup
        Exit Function
    End If

    CopyMemory HOST, lpHost, Len(HOST)
    CopyMemory dwIPAddr, HOST.hAddrList, 4

    Var tmpIPAddr(HOST.hLen) As Byte
    CopyMemory tmpIPAddr(1), dwIPAddr, HOST.hLen

    For i = 1 To HOST.hLen
        sIPAddr = sIPAddr & Str$(tmpIPAddr(i)) & "."
    Next

    GetIPAddress = Mid$(sIPAddr, 1, Len(sIPAddr) - 1)
    SocketsCleanup
End Function

'=====================================
'=
'=====================================
Declare Function GetIPHostName() As String
Function GetIPHostName() As String
    If Not SocketsInitialize Then
        GetIPHostName = ""
        Exit Function
    End If

    If Api_gethostname(sHostName, 256) = SOCKET_ERROR Then
        GetIPHostName = ""
        A% = MessageBox("", "Windowsソケットエラー。ホスト名を取得できません！", 0, 2)
```

```
        SocketsCleanup
        Exit Function
    End If

    GetIPHostName = Left$(sHostName, InStr(sHostName, Chr$(0)) - 1)
    SocketsCleanup
End Function

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Text(0).SetWindowText GetIPAddress
    Text(1).SetWindowText sHostName
End Sub

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

IPアドレス取得(II)

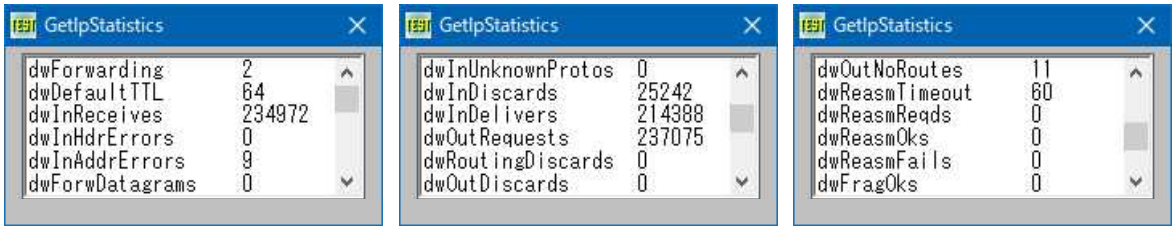
IPアドレスを取得します。
MoveMemory メモリー領域をコピー
gethostname ローカルマシンのホスト名を取得
gethostbyname インターネットホスト名に対するIPアドレスを取得
WSAStartup WinSockを初期化
WSACleanup WinSockのリソースを解放
WSAGetLastError WinSockのエラーコードを取得


```
'=======================================
'= IPAddress取得
'=    (GetIpAddress.bas)
'=======================================
Type WSADATA
    wVersion          As Integer
    wHighVersion      As Integer
    szDescription     As String * 257
    szSystemStatus    As String * 129
    iMaxSockets       As Integer
    iMaxUdpDg         As Integer
    lpVendorInfo      As Long
End Type

Type hostent
    h_name            As Long
    h_Aliases         As Long
    h_addrType        As Integer
    h_length          As Integer
    h_addr_list       As Long
End Type
```

```
' メモリの指定領域をコピー
Declare Sub MoveMemory1 Lib "kernel32" Alias "RtlMoveMemory" (Dest As Any, ByVal Source&,
ByVal length&)

' メモリの指定領域をコピー
Declare Sub MoveMemory2 Lib "kernel32" Alias "RtlMoveMemory" (Dest&, ByVal Source&, ByVal
length&)

' ローカルマシンのホスト名を取得
Declare Function Api_gethostname& Lib "wsock32" Alias "gethostname" (ByVal Name$, ByVal
namelen&)

' インターネットホスト名に対応するIPアドレスを取得
Declare Function Api_gethostbyname& Lib "wsock32" Alias "gethostbyname" (ByVal
HostName$)

' WinSockを初期化
Declare Function Api_WSAStartup& Lib "wsock32" Alias "WSAStartup" (ByVal
wVersionRequested%, lpWSAData As Any)

' WinSock のリソースを解放
Declare Function Api_WSACleanup& Lib "wsock32" Alias "WSACleanup" ()

' WinSockのエラーコードを取得
Declare Function Api_WSAGetLastError& Lib "wsock32" Alias "WSAGetLastError" ()

'=====================================
'=
'=====================================
Declare Function GetIpAddress() As String
Function GetIpAddress() As String
    Var wsa As WSADATA
    Var sName As String * 65
    Var he As hostent
    Var p As Long
    Var b(3) As byte
    Var Ret As Long

    If Api_WSAStartup(&H101, wsa) <> 0 Then exit Function

    If Api_gethostname(sName, 64) = 0 Then
        p = Api_gethostbyname(sName)
        If p <> 0 Then
            MoveMemory1 he, p, 16
            MoveMemory2 p, he.h_addr_list, 4
            MoveMemory1 b(0), p, 4
            GetIpAddress = Trim$(Str$(b(0))) & "." & Trim$(Str$(b(1))) & "." &
Trim$(Str$(b(2))) & "." & Trim$(Str$(b(3)))
        End If
    End If

    Ret = Api_WSACleanup
End Function

'---------------------------------------------------------------------

Print GetIpAddress

Stop
End
```

IPに関する統計情報を取得

IPに関する統計情報を取得します。
GetIpStatistics IP(Internet Protocol)の統計値を取得

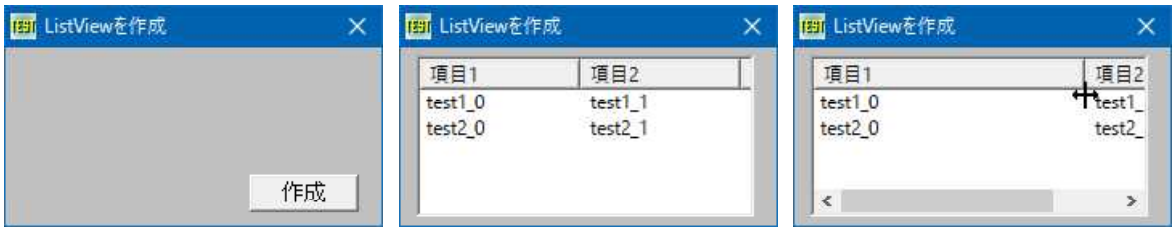



```
'=====================================
'= IPに関する統計情報を取得
'=    (GetIpStatistics.bas)
'=====================================
#include "Windows.bi"

Type MIB_IPSTATS
    dwForwarding      As Long              'IPフォワーディングのEnabled/Disabledを示す
    dwDefaultTTL      As Long              'Default TTL(Time-To-Live)
    dwInReceives      As Long              '受信したデータグラムの数
    dwInHdrErrors     As Long              'ヘッダエラーを含むデータグラムを受信した数
    dwInAddrErrors    As Long              'アドレスエラーを含むデータグラムを受信した数
    dwForwDatagrams   As Long              'フォワードしたデータグラムの数
    dwInUnknownProtos As Long              '不明なプロトコルを持つデータグラムを受信した数
    dwInDiscards      As Long              'Discardしたデータグラムの数
    dwInDelivers      As Long              '受信したデータグラムのうち、配送されたものの数
    dwOutRequests     As Long              '送信しようとしたデータグラムの数
    dwRoutingDiscards As Long              '送信されずにDiscardされたデータグラムの数
    dwOutDiscards     As Long              'Discardされた転送データグラムの数
    dwOutNoRoutes     As Long              '経路が存在せずにDiscardされたデータグラムの数
    dwReasmTimeout    As Long              'リアセンブルをあきらめるまでのタイムアウト
    dwReasmReqds      As Long              'リアセンブルを要求したデータグラムの数
    dwReasmOks        As Long              'リアセンブルが成功したデータグラムの数
    dwReasmFails      As Long              'リアセンブルが失敗したデータグラムの数
    dwFragOks         As Long              'フラグメントが成功したデータグラム数
    dwFragFails       As Long              'フラグメントが失敗したデータグラム数
    dwFragCreates     As Long              '生成されたフラグメント数
    dwNumIf           As Long              'インターフェース数
    dwNumAddr         As Long              'ローカルマシンに関連するIPアドレスの数
    dwNumRoutes       As Long              '経路表にある経路の数
End Type

' IP(Internet Protocol)の統計値を取得
Declare Function Api_GetIpStatistics& Lib "iphlpapi" Alias "GetIpStatistics" (pStats As
MIB_IPSTATS)

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

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var IpStats As MIB_IPSTATS

    Ret = Api_GetIpStatistics(IpStats)

    List1.ResetContent
    List1.AddString "dwForwarding          " & Str$(IpStats.dwForwarding)
    List1.AddString "dwDefaultTTL          " & Str$(IpStats.dwDefaultTTL)
    List1.AddString "dwInReceives          " & Str$(IpStats.dwInReceives)
    List1.AddString "dwInHdrErrors         " & Str$(IpStats.dwInHdrErrors)
    List1.AddString "dwInAddrErrors        " & Str$(IpStats.dwInAddrErrors)
    List1.AddString "dwForwDatagrams       " & Str$(IpStats.dwForwDatagrams)
    List1.AddString "dwInUnknownProtos     " & Str$(IpStats.dwInUnknownProtos)
    List1.AddString "dwInDiscards          " & Str$(IpStats.dwInDiscards)
    List1.AddString "dwInDelivers          " & Str$(IpStats.dwInDelivers)
    List1.AddString "dwOutRequests         " & Str$(IpStats.dwOutRequests)
    List1.AddString "dwRoutingDiscards     " & Str$(IpStats.dwRoutingDiscards)
    List1.AddString "dwOutDiscards         " & Str$(IpStats.dwOutDiscards)
```

```
    List1.AddString "dwOutNoRoutes          " & Str$(IpStats.dwOutNoRoutes)
    List1.AddString "dwReasmTimeout         " & Str$(IpStats.dwReasmTimeout)
    List1.AddString "dwReasmReqds           " & Str$(IpStats.dwReasmReqds)
    List1.AddString "dwReasmOks             " & Str$(IpStats.dwReasmOks)
    List1.AddString "dwReasmFails           " & Str$(IpStats.dwReasmFails)
    List1.AddString "dwFragOks              " & Str$(IpStats.dwFragOks)
    List1.AddString "dwFragFails            " & Str$(IpStats.dwFragFails)
    List1.AddString "dwFragCreates          " & Str$(IpStats.dwFragCreates)
    List1.AddString "dwNumIf                " & Str$(IpStats.dwNumIf)
    List1.AddString "dwNumAddr              " & Str$(IpStats.dwNumAddr)
    List1.AddString "dwNumRoutes            " & Str$(IpStats.dwNumRoutes)
End Sub

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

ListViewをコードで作成

ListViewを作成します。
CreateWindowEx ウインドウ(コントロール)を作成
DestroyWindow CreateWindowExの解放
SendMessage ウィンドウにメッセージを送信




```
'=====================================
'= ListViewをコードで作成
'=    (CreateWindowEx6.bas)
'=====================================
#include "Windows.bi"

Type LVITEM
    mask       As Long
    iItem      As Long
    iSubItem   As Long
    state      As Long
    stateMask  As Long
    pszText    As Long
    cchTextMax As Long
    iImage     As Long
    lParam     As Long
    iIndent    As Long
End Type

Type LVCOLUMN
    mask       As Long
    fmt        As Long
    cx         As Long
    pszText    As Long
    cchTextMax As Long
    iSubitem   As Long
End Type

#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
```

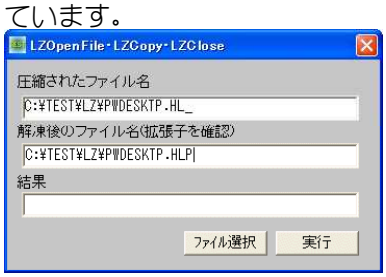
```
#define WS_EX_CLIENTEDGE &H200              '縁が沈んで見える境界線を持つウィンドウを指定
#define WS_VISIBLE &H10000000               '可視状態のウィンドウを作成する
#define LVS_LIST &H3                        'リストビューを指定
#define LVS_REPORT &H1                      'レポートビューを指定
#define LVS_AUTOARRANGE &H100               'アイコンビュー及び小さなアイコンビューで、アイコンが自
                                            動的整列
#define LVM_FIRST &H1000                    '
#define LVM_INSERTCOLUMN &H101B             '新しいカラム(列)を挿入
#define LVM_SETIMAGELIST &H1003             'イメージリストの割り当て
#define LVM_GETIMAGELIST &H1002
#define LVSIL_SMALL 1                       '小さいアイコン
#define LVSIL_NORMAL 0                      '大きいアイコン
#define LVCF_TEXT &H4                       'LVCOLUMNメンバ(pszText)
#define LVCF_WIDTH &H2                      'LVCOLUMNメンバ(cx)
#define LVIF_TEXT &H1
#define LVIF_IMAGE &H2
#define LVIF_PARAM &H4
#define LVIF_STATE &H8
#define LVM_INSERTITEMA &H1007              '(LVM_FIRST + 7)
#define LVM_GETITEMCOUNT &H1004             '(LVM_FIRST + 4)
#define LVM_SETITEMA &H1006                 '(LVM_FIRST + 6)
#define LVM_SETITEMTEXTA (&H1000 Or 46)     '(LVM_FIRST + 46)

' ウインドウ(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&)

' 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_SendMessage& Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal
wMsg&, ByVal wParam&, lParam As Any)

Var Shared Button1 As Object

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

Var Shared hWndLV As Long
Var Shared Style As Long

'=====================================
'=
'=====================================
Declare Function AddItem(hwnd As Long, Index As Long, Buffer As String)
Function AddItem(hwnd As Long, Index As Long, Buffer As String)
    Var lvi As LVITEM
    Var test As Long

    lvi.mask = LVIF_TEXT Or LVIF_IMAGE Or LVIF_PARAM Or LVIF_STATE
    lvi.state = 0
    lvi.stateMask = 0
    lvi.pszText = StrAdr(Buffer & Chr$(0))
    lvi.iImage = 0
    lvi.iItem = Api_SendMessage(hWndLV, LVM_GETITEMCOUNT, 0, 0)

    AddItem = Api_SendMessage(hWndLV, LVM_INSERTITEMA, Index, lvi)
End Function

'=====================================
'=
'=====================================
Declare Function AddSubItem(hwnd As Long, Index As Long, SubIndex As Long, Buffer As
String)
Function AddSubItem(hwnd As Long, Index As Long, SubIndex As Long, Buffer As String)
    Var lvi As LVITEM

    lvi.mask = LVIF_TEXT Or LVIF_IMAGE Or LVIF_PARAM Or LVIF_STATE
    lvi.state = 0
```

```
    lvi.stateMask = 0
    lvi.pszText = StrAdr(Buffer & Chr$(0))
    lvi.iItem = Api_SendMessage(hWndLV, LVM_GETITEMCOUNT, 0, 0)
    lvi.iSubItem = SubIndex

    AddSubItem = Api_SendMessage(hwnd, LVM_SETITEMTEXTA, Index, lvi)
End Function

'=====================================
'=
'=====================================
Declare Function AddColumn(hwnd As Long, txt As String)
Function AddColumn(hwnd As Long, txt As String)
    Var lvc As LVCOLUMN
    Var Col As Long

    lvc.mask = LVCF_TEXT Or LVCF_WIDTH
    lvc.pszText = StrAdr(txt & Chr$(0))
    lvc.fmt = 2
    lvc.cx = 100

    AddColumn = Api_SendMessage(hwnd, LVM_INSERTCOLUMN, Col, lvc)
End Function

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

    Style = WS_CHILD Or WS_VISIBLE Or LVS_REPORT Or LVS_AUTOARRANGE

    hWndLV = Api_CreateWindowEx(WS_EX_CLIENTEDGE, "SysListView32", ByVal 0, Style, 10, 5,
210, 100, GethWnd, 0, GetInst, ByVal 0)

    Ret = AddColumn(hWndLV, "項目2")
    Ret = AddColumn(hWndLV, "項目1")

    Ret = AddItem(hWndLV, 0, "test1_0")
    Ret = AddItem(hWndLV, 1, "test2_0")

    Ret = AddSubItem(hWndLV, 0, 1, "test1_1")
    Ret = AddSubItem(hWndLV, 1, 1, "test2_1")

    Button1.DestroyWindow
End Sub

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

    Ret = Api_DestroyWindow(hWndLV)
End Sub

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

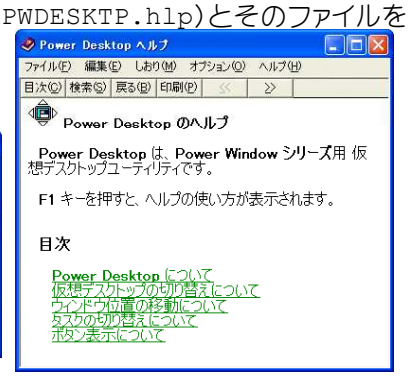
Lz形式で圧縮されたファイルの解凍

あまり見かけなくなりましたが、COMPRESS.EXEで圧縮されたファイル（拡張子の後尾に「 _ 」が付く）を解凍します。
LZOpenFile 解凍するファイルと、解凍後のファイルのハンドルを取得
LZCopy 元ファイルを、解凍後ファイルにコピー
LZClose ファイルを閉じる
PathRemoveExtension パス文字列から拡張子を取り除く

例では、あらかじめC:¥TEST¥LZに用意した圧縮ファイルPWDESKTO.HL_を選択し、解凍後の拡張子をHLPと設定しています。



解凍されたファイルの存在確認（PWDESKTP.hlp）とそのファイルを開いたところ




```
'=======================================
'= LZ形式で圧縮されたファイルの解凍
'=    (LZOpenFile.bas)
'=======================================
#include "Windows.bi"

Type OFSTRUCT
    cBytes          As byte
    fFixedDisk      As byte
    nErrCode        As Integer
    Reserved1       As Integer
    Reserved2       As Integer
    szPathName      As String * 128
End Type

' 解凍するファイルと、解凍後のファイルのハンドルを取得
Declare Function Api_LZOpenFile& Lib "lz32" Alias "LZOpenFileA" (ByVal lpszFile$, lpOf As
OFSTRUCT, ByVal style&)

' 元ファイルを、解凍後ファイルにコピー
Declare Function Api_LZCopy& Lib "lz32" Alias "LZCopy" (ByVal hfSource&, ByVal hfDest&)

' ファイルを閉じる
Declare Sub Api_LZClose Lib "lz32" Alias "LZClose" (ByVal hfFile&)

' パス文字列から拡張子を取り除く関数
Declare Sub Api_PathRemoveExtension Lib "shlwapi" Alias "PathRemoveExtensionA" (ByVal
pszPath$)

#define OF_READ &H0                         'ファイルを読みとり専用で開く
#define OF_CREATE &H1000                    '新しいファイルを作成する
#define LZERROR_BADINHANDLE (-1)            'コピー元ファイルのハンドルが無効
#define LZERROR_BADOUTHANDLE (-2)           'コピー先ファイルのハンドルが無効
#define LZERROR_BADVALUE (-7)               '入力パラメータの 1 つが無効
#define LZERROR_GLOBLOCK (-6)               'LZファイルハンドルをロックできない
```

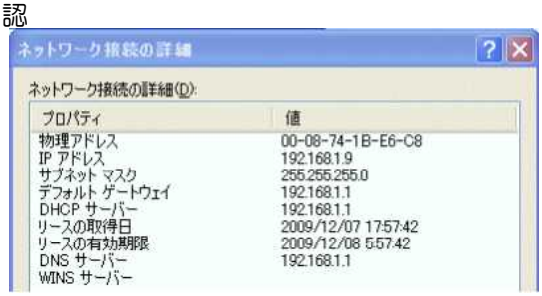
```
#define LZERROR_PUBLICLOC (-5)              '
#define LZERROR_READ (-3)                   'コピー元ファイルの形式が無効
#define LZERROR_UNKNOWNALG (-8)             'コピー元ファイルは認識できない形式で圧縮されている
#define LZERROR_WRITE (-4)                  'コピー先ディスクの容量が足りない

Var Shared Text(1) As Object
Var Shared Edit1 As Object

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

Var Shared FileName As String

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    FileName = WinOpenDlg("ファイルのオープン", "*.*", "全てのﾌｧｲﾙ(*.*)", 0)
    If FileName <> Chr$(&H1B) Then
        Text(0).SetWindowText FileName
        dName$ = FileName
        Api_PathRemoveExtension dName$
        Edit1.SetWindowText dName$
        Edit1.SetSelText Len(dName$), Len(dName$)
        Edit1.SetFocus
    End If
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var SourceStruct As OFSTRUCT
    Var DestStruct As OFSTRUCT
    Var hSource As Long
    Var hDest As Long
    Var Ret As Long

    'ソースと新しいファイルを開く
    hSource = Api_LZOpenFile(FileName, SourceStruct, OF_READ)
    hDest = Api_LZOpenFile(Edit1.GetWindowText, DestStruct, OF_CREATE)

    'ファイルをコピーする
    Ret = Api_LZCopy(hSource, hDest)

    'ファイルを閉じる
    Api_LZClose hSource
    Api_LZClose hDest

    'エラーチェックと成功
    Select Case Ret
        Case LZERROR_BADINHANDLE
            Text(1).SetWindowText "LZERROR_BADINHANDLE"
        Case LZERROR_BADOUTHANDLE
            Text(1).SetWindowText "LZERROR_BADOUTHANDLE"
        Case LZERROR_BADVALUE
            Text(1).SetWindowText "LZERROR_BADVALUE"
        Case LZERROR_GLOBLOCK
            Text(1).SetWindowText "LZERROR_GLOBLOCK"
        Case LZERROR_PUBLICLOC
            Text(1).SetWindowText "LZERROR_PUBLICLOC"
        Case LZERROR_READ
            Text(1).SetWindowText "LZERROR_READ"
        Case LZERROR_UNKNOWNALG
            Text(1).SetWindowText "LZERROR_UNKNOWNALG"
```

```
        Case LZERROR_WRITE
            Text(1).SetWindowText "LZERROR_WRITE"
        Case Else
            Text(1).SetWindowText "Success !"
    End Select
End Sub

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

MACアドレスを取得(ＩＰ)

MAC(Media Access Control)アドレスを取得します。🍎ではありません‥
Netbios 指定されたネットワークコントロールブロック(NCB)を解釈実行
GetProcessHeap 呼び出し側プロセスのヒープのハンドルを取得
RtlMoveMemory メモリーのブロックのコピー
HeapAlloc 確保したメモリー領域を割り当て
HeapFree HeapReAlloc機能により割り当てられたメモリーブロックを解放

Dell(Windows XP Pro)での確認



VersaPro(Windows XP Pro)での確認
Hp(Windows 7)での確認(VB6でも同じ結果でした)

ネットワーク接続→ローカルエリア接続→ローカルエリア接続の状態→サポート→詳細と進んでいくと見られます。

```
'=====================================
'= MACアドレスを取得
'=    (Netbios.bas)
'=    VB6 からネットワーク アダプタ アドレスを取得する方法
'=====================================
#include "Windows.bi"

#define NCBASTAT &H33                    '
#define NCBNAMSZ 16                      '
#define NCBRESET &H32                    '
```

```
#define HEAP_ZERO_MEMORY &H8                '
#define HEAP_GENERATE_EXCEPTIONS &H4        '

Type NET_CONTROL_BLOCK                      'NCB
    ncb_command       As Byte
    ncb_retcode       As Byte
    ncb_lsn           As Byte
    ncb_num           As Byte
    ncb_buffer        As Long
    ncb_length        As Integer
    ncb_callname      As String * NCBNAMSZ
    ncb_name          As String * NCBNAMSZ
    ncb_rto           As Byte
    ncb_sto           As Byte
    ncb_post          As Long
    ncb_lana_num      As Byte
    ncb_cmd_cplt      As Byte
    ncb_reserve(9)    As Byte
    ncb_event         As Long
End Type

Type ADAPTER_STATUS
    adapter_address(5) As Byte
    rev_major         As Byte
    reserved0         As Byte
    adapter_Type      As Byte
    rev_minor         As Byte
    duration          As Integer
    frmr_recv         As Integer
    frmr_xmit         As Integer
    iframe_recv_err   As Integer
    xmit_aborts       As Integer
    xmit_success      As Long
    recv_success      As Long
    iframe_xmit_err   As Integer
    recv_buff_unavail As Integer
    t1_timeouts       As Integer
    ti_timeouts       As Integer
    Reserved1         As Long
    free_ncbs         As Integer
    max_cfg_ncbs      As Integer
    max_ncbs          As Integer
    xmit_buf_unavail  As Integer
    max_dgram_size    As Integer
    pEnding_sess      As Integer
    max_cfg_sess      As Integer
    max_sess          As Integer
    max_sess_pkt_size As Integer
    name_count        As Integer
End Type

Type NAME_BUFFER
    bname             As String * NCBNAMSZ
    name_num          As Integer
    name_flags        As Integer
End Type

Type ASTAT
    adapt             As ADAPTER_STATUS
    NameBuff(30)      As NAME_BUFFER
End Type

' 指定されたネットワークコントロールブロック(NCB)を解釈実行
Declare Function Api_Netbios& Lib "netapi32" Alias "Netbios" (pncb As NET_CONTROL_BLOCK)

' 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" (hpvDest As Any, ByVal
hpvSource&, ByVal cbCopy&)
```

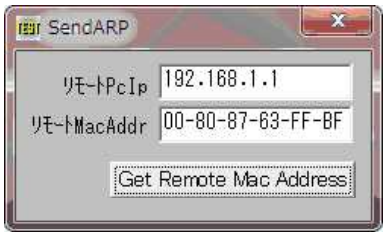
```
' 呼び出し側プロセスのヒープのハンドルを取得
Declare Function Api_GetProcessHeap& Lib "Kernel32" Alias "GetProcessHeap" ()

' 確保したメモリー領域を割り当て
Declare Function Api_HeapAlloc& Lib "Kernel32" Alias "HeapAlloc" (ByVal hHeap&, ByVal
dwFlags&, ByVal dwBytes&)

' HeapReAlloc機能により割り当てられたメモリーブロックを解放
Declare Function Api_HeapFree& Lib "Kernel32" Alias "HeapFree" (ByVal hHeap&, ByVal
dwFlags&, lpMem 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 Function GetMACAddress$ ()
Function GetMACAddress$()
    Var tmp As String
    Var pASTAT As Long
    Var NCB As NET_CONTROL_BLOCK
    Var AST As ASTAT
    Var Ret As Long

    NCB.ncb_command = NCBRESET

    Ret = Api_Netbios(NCB)

    NCB.ncb_callname = "*               "
    NCB.ncb_command = NCBASTAT
    NCB.ncb_lana_num = 0
    NCB.ncb_length = Len(AST)

    pASTAT = Api_HeapAlloc(Api_GetProcessHeap(), HEAP_GENERATE_EXCEPTIONS Or
HEAP_ZERO_MEMORY, NCB.ncb_length)

    If pASTAT = 0 Then
        A% = MessageBox("", "メモリ取得に失敗しました", 0, 2)
        Exit Function
    End If

    NCB.ncb_buffer = pASTAT
    Ret = Api_Netbios(NCB)

    CopyMemory AST, NCB.ncb_buffer, Len(AST)

    tmp = Right$("0" & Hex$(AST.adapt.adapter_address(0)), 2) & " " & Right$("0" &
Hex$(AST.adapt.adapter_address(1)), 2) & " " & Right$("0" &
Hex$(AST.adapt.adapter_address(2)),2)
    tmp = tmp & " " & Right$("0" & Hex$(AST.adapt.adapter_address(3)), 2) & " " & Right$("0"
& Hex$(AST.adapt.adapter_address(4)), 2) & " " & Right$("0" &
Hex$(AST.adapt.adapter_address(5)), 2)

    Ret = Api_HeapFree(Api_GetProcessHeap(), 0, pASTAT)
    GetMACAddress = tmp
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Text2.SetWindowText GetMACAddress()
End Sub
```

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

MACアドレス取得(II)

Addres Resolution ProtocolによりIPアドレスからMACアドレスを取得します。
inet_addr 文字列を32ビットのバイナリ値に変換
CopyMemory ある位置から別の位置にメモリブロックを移動
SendARP ARPによるMACアドレス取得

コマンドプロンプトによるLAN接続されているPCのIPアドレスおよびMACアドレスの確認




```
'=====================================
'= MACアドレス取得(II)
'=    (SendARP.bas)
'=====================================
#include "Windows.bi"

#define NO_ERROR 0

' 文字列を32ビットのバイナリ値に変換
Declare Function Api_inet_addr& Lib "wsock32" Alias "inet_addr" (ByVal s$)

' 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&)

' ARP(Address Resolution Protocol)によるMACアドレス取得
Declare Function Api_SendARP& Lib "iphlpapi" Alias "SendARP" (ByVal DestIP&, ByVal
SrcIP&, MacAddr&, PhyAddrLen&)
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
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Function GetRemoteMacAddress(sRemoteIP As String, sRemoteMacAddress As String)
As Integer
Function GetRemoteMacAddress(sRemoteIP As String, sRemoteMacAddress As String) As
Integer
    Var RemoteIP As Long
    Var MacAddr As Long
    Var PhyAddrLen As Long
    Var cnt As Long
```

```
    Var tmp As String

    RemoteIP = Api_inet_addr(sRemoteIP)

    If RemoteIP <> 0 Then
        PhyAddrLen = 6

        If Api_SendARP(RemoteIP, 0, MacAddr, PhyAddrLen) = NO_ERROR Then
            If MacAddr <> 0 And PhyAddrLen <> 0 Then
                Var bMacAddr(PhyAddrLen - 1) As Byte
                CopyMemory bMacAddr(0), MacAddr, ByVal PhyAddrLen

                For cnt = 0 To PhyAddrLen - 1
                    If bMacAddr(cnt) = 0 Then
                        tmp = tmp & "00-"
                    Else
                        tmp = tmp & Right$("00" & Hex$(bMacAddr(cnt)), 2) & "-"
                    End If
                Next

                If Len(tmp) > 0 Then
                    sRemoteMacAddress = Left$(tmp, Len(tmp) - 1)
                    GetRemoteMacAddress = True
                End If
                Exit Function
            Else
                GetRemoteMacAddress = False
            End If
        Else
            GetRemoteMacAddress = False
        End If
    Else
        GetRemoteMacAddress = False
    End If
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var sRemoteMacAddress As String

    If Len(Edit(0).GetWindowText) > 0 Then
        If GetRemoteMacAddress(Edit(0).GetWindowText, sRemoteMacAddress) Then
            Edit(1).SetWindowText sRemoteMacAddress
        Else
            Edit(1).SetWindowText "(取得失敗)"
        End If
    End If
End Sub

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

MIDI出力デバイス名を列挙

インストールされているMIDIデバイス名を列挙します。
midiOutGetDevCaps MIDIデバイスのデバイス名を取得
midiOutGetNumDevs MIDIデバイス数を取得

確認

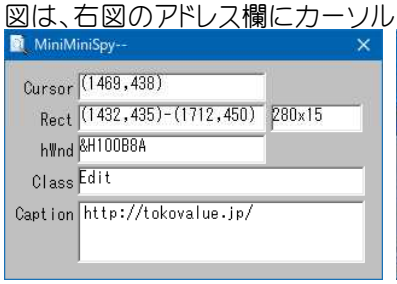

```
'=====================================
'= MIDI出力デバイス名を列挙
'=    (midiOutGetDevCaps.bas)
'=====================================

#define MAXPNAMELEN 32

Type MIDIOUTCAPS
    wMid              As Integer
    wPid              As Integer
    vDriverVersion    As Long
    szPname           As String * MAXPNAMELEN
    wTechnology       As Integer
    wVoices           As Integer
    wNotes            As Integer
    wChannelMask      As Integer
    dwSupport  As Long
End Type

' MIDI出力デバイスのデバイス名を取得
Declare Function Api_midiOutGetDevCaps& Lib "winmm" Alias "midiOutGetDevCapsA" (ByVal
uDeviceID&, lpCaps As MIDIOUTCAPS, ByVal uSize&)

' MIDI 出力デバイス数を取得
Declare Function Api_midiOutGetNumDevs& Lib "winmm" Alias "midiOutGetNumDevs" ()

var mc as MIDIOUTCAPS
var cnt As Long
var Ret As Long

'インストールされているMIDIデバイスの数を取得
Print "MIDIデバイスの数:" & Str$(Api_midiOutGetNumDevs)

For cnt = 0 To Api_midiOutGetNumDevs - 1

    'デバイス名と機能を取得
    Ret = Api_midiOutGetDevCaps(cnt, mc, Len(mc))
    Print "デバイス名" & Str$(cnt + 1) & ":" & Left$(mc.szPname, InStr(mc.szPname, Chr$(0)) -
1)
Next

Stop
End
```

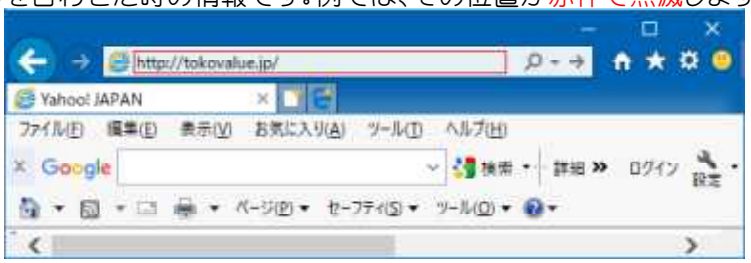
MiniMiniSpy--

ウインドウクラスの情報を取得します。名付けて MiniMiniSpy--（かくいう私は、本物のSpy++を拝んだことがありません(^^;)

GetWindowRect ウィンドウの座標をスクリーン座標系で取得
GetCursorPos カーソルの現在のスクリーン座標の取得
WindowFromPoint 指定の座標位置にあるウィンドウハンドルを取得
GetClassName ウィンドウのクラス名を取得
GetWindowText ウインドウのタイトル文字列を取得
SendMessage ウィンドウにメッセージを送信
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
GetWindowDC ウィンドウ全体のデバイスコンテキストを取得
GetPixel 指定された座標のピクセルのRGB値を取得
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
CreatePenIndirect LOGPEN構造体を定義して論理ペンを作成
DeleteObject システムリソースを解放

カーソル位置のクラス名を取得し、そのハンドル・矩形領域値・そのサイズ値を表示するとともにその矩形領域枠を指定色（現在・赤）で描画します。
図は、右図のアドレス欄にカーソルを合わせた時の情報です。例では、その位置が赤枠で点滅します。





※経緯
　最初は、Form2を用意し、そのサイズを選択矩形領域に合わせ、内部を透明処理しました。これでは動作が遅いため、デバイスコンテキストに枠を描画することにしました。
　Focusの移動があった場合、その枠を消さなければならないので、領域確保したときの左上座標色を記憶しておき、枠を描画し適当なWait処理後、その枠を記憶した元の色で再描画しています。

フォームレイアウトを変えただけです

MiniMiniSpy--は、領域を赤枠で描画していますが、DrawFocusRectで点線で描画する方法もあります。 ※参照: フォーカスを得たとき点線の枠を描画

```
'=======================================
'= ウインドウクラス情報取得
'=    (MiniMiniSpy--.bas)
'=======================================
#include "Windows.bi"
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

Type LOGPEN
    lopnStyle  As Long
    lopnWidth  As POINTAPI
    lopnColor  As Long
End Type

' 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)

' カーソルの現在のスクリーン座標の取得
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&)

' ウィンドウのクラス名を取得する関数の宣言
Declare Function Api_GetClassName& Lib "user32" Alias "GetClassNameA" (ByVal hWnd&,
ByVal lpClassName$, ByVal nMaxCount&)

' 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&)

' 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)

' 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&)

' 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&)

' LOGPEN構造体を定義して論理ペンを作成
Declare Function Api_CreatePenIndirect& Lib "gdi32" Alias "CreatePenIndirect" (lpLogPen
As LOGPEN)

' 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&)

#define WM_GETTEXT &HD                          'コントロールのキャプション・テキストをバッファにコピー
Var Shared Timer1 As Object
Var Shared Text(9) As Object
Var Shared Edit1 As Object

Timer1.Attach GetDlgItem("Timer1")
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

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

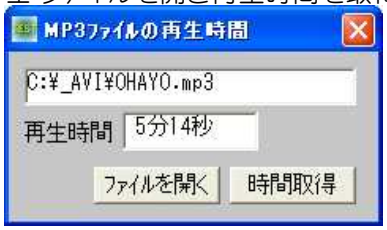
```
        TrimNull = Left$(item, ePos - 1)
    Else
        TrimNull = item
    End If
End Function

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Timer1.SetInterval 50
    Timer1.Enable -1
End sub

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var pa As POINTAPI
    Var rc As RECT
    Var hWnd As Long
    Var Buff As String * 128
    Var Caption As String
    Var hDC As Long
    Var hNewPen As Long
    Var hOldPen As Long
    Var NewPen As LOGPEN
    Var ColorMe As Long
    Var rgbRed As Long
    Var rgbGreen As Long
    Var rgbBlue As Long
    Var Ret As Long

    'カーソル位置のスクリーン座標を取得
    Ret = Api_GetCursorPos(pa)

    '座標を含むウィンドウのハンドルを取得
    hWnd = Api_WindowFromPoint(pa.x, pa.y)

    '矩形領域を取得
    Ret = Api_GetWindowRect(hWnd, rc)

    'デバイスコンテキストを取得
    hDC = Api_GetWindowDC(hWnd)

    '選択したデバイスの左上の色を記憶
    ColorMe = Api_GetPixel(hDC, 0, 0)

    'RGBに分解
    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)

    If hWnd <> 0 Then                          'ウィンドウのハンドルを取得できたとき

        NewPen.lopnColor = Rgb(255, 0, 0)
        Gosub *FrameDraw
        Wait 10

        'ペンの作成
        NewPen.lopnColor = ColorMe
        Gosub *FrameDraw

        'カーソル座標を表示
        Text(4).SetWindowText "(" & Trim$(Str$(pa.x)) & "," & Trim$(Str$(pa.y)) & ")"
```

```
        '矩形座標を表示
        Text(5).SetWindowText "(" & Trim$(Str$(rc.Left)) & "," & Trim$(Str$(rc.Top)) &
")-(" & Trim$(Str$(rc.Right)) & "," & Trim$(Str$(rc.Bottom)) & ")"

        '矩形サイズを表示
        Text(6).SetWindowText Trim$(Str$(rc.Right - rc.Left)) & "x" &
Trim$(Str$(rc.Bottom - rc.Top))

        'ウィンドウのハンドルを表示
        Text(7).SetWindowText "&&H" & Hex$(hWnd)

        'ウィンドウのクラス名を取得
        Ret = Api_GetClassName(hWnd, Buff, Len(Buff))

        Text(8).SetWindowText TrimNull(Buff)

        'キャプション取得
        Caption = Space$(250)
        Ret = Api_GetWindowText(hWnd, Caption, Len(Caption))
        Ret = Api_SendMessage(hWnd, WM_GETTEXT, Len(Caption), Caption)
        Edit1.SetWindowText TrimNull(Caption)
    End If
    Ret = Api_DeleteObject(hNewPen)
    Exit Sub

*FrameDraw

    'ペンの作成
    NewPen.lopnWidth.x = 1
    hNewPen = Api_CreatePenIndirect(NewPen)

    'ペンを持ち替える
    hOldPen = Api_SelectObject(hDC, hNewPen)
    Ret = Api_MoveToEx(hDC, 0, 0, ByVal 0)                          '┌    左上(始点)
    Ret = Api_LineTo(hDC, rc.Right - rc.Left - 1, 0)                '  ―  右上
    Ret = Api_LineTo(hDC, rc.Right - rc.Left - 1, rc.Bottom - rc.Top - 1)
                                                                    '    | 右下
    Ret = Api_LineTo(hDC, 0, rc.Bottom - rc.Top - 1)                ' ―   左下
    Ret = Api_LineTo(hDC, 0, 0)                                     '|    左上(終点)
    hNewPen = Api_SelectObject(hWnd, hOldPen)
    Return
End Sub

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

MP3ファイルの再生時間を取得

mciSendString 文字列を MCI(Multimedia Control Interface) に送信
GetShortPathName ファイルの短い形式のパス名を取得

左:ファイルを開き再生時間を取得　右:確認



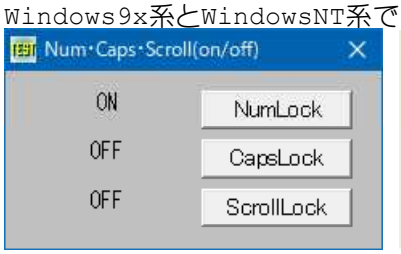
```
'=====================================
'= MP3ファイルの再生時間を取得
'=    (mciSendString2.bas)
'=====================================
#include "Windows.bi"

' 文字列を MCI に送信
Declare Function Api_mciSendString& Lib "winmm" Alias "mciSendStringA" (ByVal
lpstrCommand$, ByVal lpstrReturnString$, ByVal uReturnLength&, ByVal hwndCallback&)

' 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&)

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

Var Shared FileName As String

'=====================================
'=
'=====================================
Declare Function GetMP3Length(FileName As String) As Long
Function GetMP3Length(FileName As String) As Long
    Var Buffer As String
    Var Length As String
    Var Ret As Long
    Buffer = space$(255)
    Ret = Api_GetShortPathName(FileName, Buffer, Len(Buffer))

    If Ret <> 0 Then
        FileName = Left$(Buffer, InStr(Buffer, Chr$(0)) - 1)
    End If

    Ret = Api_mciSendString("open " & FileName & " Type MPEGVideo Alias mp3audio", ByVal 0,
0, 0)

    Length = space$(256)
    Ret = Api_mciSendString("status mp3audio length", Length, Len(Length), 0)

    Ret = Api_mciSendString("close mp3audio", ByVal 0, 0, 0)

    GetMP3Length = Val(Length)
End Function

'=====================================
'=
'=====================================
Declare Function FormatTime(ByVal fTime As Long) As String
Function FormatTime(ByVal fTime As Long) As String
    Var Min As Integer
    Var Sec As Integer

    Sec = int(fTime / 1000)
    Min = int(Sec / 60)
    Sec = Sec - (Min * 60)

    FormatTime = Format$(Min, "##") & "分" & Format$(Sec, "##") & "秒"
End Function

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

    FileName = WinOpenDlg("ファイルのオープン","C:¥*.mp3","MP3ﾌｧｲﾙ(*.mp3)", 0)
```

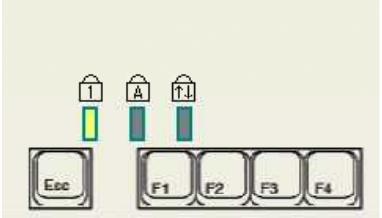
```
    If FileName <> Chr$(&H1B) Then
        Text(0).SetWindowText FileName
    End If
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()

    Text(1).SetWindowText FormatTime(GetMP3Length(FileName))
End Sub

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

NumLock･CapsLock･ScrollLockのオン･オフ(Ⅰ)

NumLock･CapsLock･ScrollLockをON/OFFさせます。
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
keybd_event 特殊キーの状態を設定
GetKeyboardState 仮想キーボードのキーの状態を取得
SetKeyboardState キーボードの仮想キーの状態を設定

Windows9x系とWindowsNT系では異なるためOSバージョンを取得し分けて操作しています。





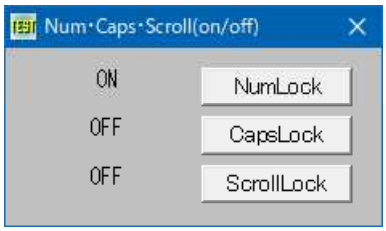

```
'=====================================
'= CapsLock･NumLock･ScrollLockのオン･オフ
'=    (CapsLock.bas)
'=====================================
#include "Windows.bi"

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)

' 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&)

' 仮想キーボードのキーの状態を取得
Declare Function Api_GetKeyboardState& Lib "user32" Alias "GetKeyboardState" (pbKeyState
As Byte)
```

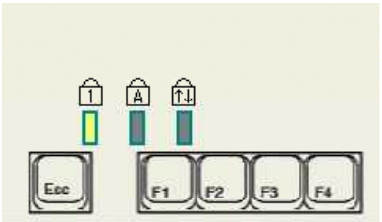
```
' キーボードの仮想キーの状態を設定
Declare Function Api_SetKeyboardState& Lib "user32" Alias "SetKeyboardState"
(lppbKeyState As Byte)

#define VK_NUMLOCK &H90                    '[NumLock]
#define VK_CAPITAL &H14                    '[Caps Lock]
#define VK_SCROLL &H91                     '[Scroll]
#define KEYEVENTF_KEYUP &H2                'キーを放す
#define KEYEVENTF_EXTENDEDKEY &H1          'スキャンコードにプリフィックスバイト0xE0(224)を付加
#define VER_PLATFORM_WIN32_NT 2            'WINDOWSNT、2000、XP
#define VER_PLATFORM_WIN32_WINDOWS 1       'WINDOWS9x

Var Shared Text(2) As Object
Var Shared Button(2) As Object

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 State(2) As Integer
Var Shared VK(2) As Long
Var Shared NumLockState As Integer
Var Shared CapsLockState As Integer
Var Shared ScrollLockState As Integer
Var Shared Index As Integer

'=====================================
'= ScrollLock
'=====================================
Declare Sub Lock_Toggle ()
Sub Lock_Toggle()
    Var OsVer As OSVERSIONINFO
    Var keys(255) As Byte
    Var Ret As Long

    OsVer.dwOSVersionInfoSize = Len(OsVer)
    Ret = Api_GetVersionEx(OsVer)
    Ret = Api_GetKeyboardState(keys(0))

    State(Index) = keys(VK(Index))

    If State(Index) = 0 Then                'Lock(OFF→ON)
        If OsVer.dwPlatformId = VER_PLATFORM_WIN32_WINDOWS Then  'Win9x
            keys(VK(Index)) = 1
            Ret = Api_SetKeyboardState(keys(0))
        Else If OsVer.dwPlatformId = VER_PLATFORM_WIN32_NT Then  'WinNT
            Api_keybd_event VK(Index), &H45, KEYEVENTF_EXTENDEDKEY Or 0, 0
            Api_keybd_event VK(Index), &H45, KEYEVENTF_EXTENDEDKEY Or KEYEVENTF_KEYUP, 0
        End If
        Text(Index).SetWindowText "ON"
    Else                                                'Lock(ON→OFF)
        If OsVer.dwPlatformId = VER_PLATFORM_WIN32_WINDOWS Then  'Win9x
            keys(VK(Index)) = 0
            Ret = Api_SetKeyboardState(keys(0))
        Else If OsVer.dwPlatformId = VER_PLATFORM_WIN32_NT Then  'WinNT
            Api_keybd_event VK(Index), &H45, KEYEVENTF_EXTENDEDKEY Or 0, 0
            Api_keybd_event VK(Index), &H45, KEYEVENTF_EXTENDEDKEY Or KEYEVENTF_KEYUP, 0
        End If
        Text(Index).SetWindowText "OFF"
    End If
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    VK(0) = VK_NUMLOCK
    VK(1) = VK_CAPITAL
```

```
    VK(2) = VK_SCROLL

    State(0) = NumLockState
    State(1) = CapsLockState
    State(2) = ScrollLockState
End Sub

'=====================================
'= NumLock
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Index = 0
    Lock_Toggle
End Sub

'=====================================
'= CapsLock
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Index = 1
    Lock_Toggle
End Sub

'=====================================
'= ScrollLock
'=====================================
Declare Sub Button3_on edecl ()
Sub Button3_on()
    Index = 2
    Lock_Toggle
End Sub

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

NumLock･CapsLock･ScrollLockのオン･オフ(II)

NumLock･CapsLock･ScrollLockをON/OFFさせます。WindowsNT系以降
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
GetKeyboardState 仮想キーボードのキーの状態を取得






```
'=====================================
'= NumLock･CapsLock･ScrollLockのオン･オフ(II)
'=     (MapVirtualKey.bas)WindowsNT以降
'=====================================
#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&)

' 仮想キーボードのキーの状態を取得
Declare Function Api_GetKeyboardState& Lib "user32" Alias "GetKeyboardState" (pbKeyState
As byte)

#define KEYEVENTF_EXTENDEDKEY &H1          'スキャンコードにプリフィックスバイト0xE0(224)を付加
#define KEYEVENTF_KEYUP &H2                'キーを放す
#define VK_SCROLL &H91                     '[Scroll]
#define VK_NUMLOCK &H90                    '[NumLock]
#define VK_CAPITAL &H14                    '[Caps Lock]

Var Shared Text(2) As Object
Var Shared Button(2) As Object

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

'=====================================
'= NumLock
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var state(255) As Byte
    Var Ret As Long

    Api_keybd_event VK_NUMLOCK, Api_MapVirtualKey(VK_NUMLOCK, 0), KEYEVENTF_EXTENDEDKEY
Or 0, 0

    Api_keybd_event VK_NUMLOCK, Api_MapVirtualKey(VK_NUMLOCK, 0), KEYEVENTF_EXTENDEDKEY
Or KEYEVENTF_KEYUP, 0

    Ret = Api_GetKeyboardState(state(0))
    If state(VK_NUMLOCK) And &H1 Then
        Text(0).SetWindowText "OFF"
    Else
        Text(0).SetWindowText "ON"
    End If
End Sub

'=====================================
'= CapsLock
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var state(255) As Byte
    Var Ret As Long

    Api_keybd_event VK_CAPITAL, Api_MapVirtualKey(VK_CAPITAL, 0), KEYEVENTF_EXTENDEDKEY
Or 0, 0

    Api_keybd_event VK_CAPITAL, Api_MapVirtualKey(VK_CAPITAL, 0), KEYEVENTF_EXTENDEDKEY
Or KEYEVENTF_KEYUP, 0

    Ret = Api_GetKeyboardState(state(0))
    If state(VK_CAPITAL) And &H1 Then
        Text(1).SetWindowText "OFF"
    Else
        Text(1).SetWindowText "ON"
    End If
End Sub
```

```
'====================================
'= ScrollLock
'====================================
Declare Sub Button3_on edecl ()
Sub Button3_on()
    Var state(255) As Byte
    Var Ret As Long

    Api_keybd_event VK_SCROLL, Api_MapVirtualKey(VK_SCROLL, 0), KEYEVENTF_EXTENDEDKEY
Or 0, 0

    Api_keybd_event VK_SCROLL, Api_MapVirtualKey(VK_SCROLL, 0), KEYEVENTF_EXTENDEDKEY
Or KEYEVENTF_KEYUP, 0

    Ret = Api_GetKeyboardState(state(0))
    If state(VK_SCROLL) And &H1 Then
        Text(2).SetWindowText "OFF"
    Else
        Text(2).SetWindowText "ON"
    End If
End Sub

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

OEMコードページの識別子を取得

GetOEMCP OEMコードページの識別子を取得


```
'====================================
'= OEMコードページの識別子を取得
'=    (GetOEMCP.bas)
'====================================
#include "Windows.bi"

' システムで現在有効になっている OEM コードページの識別子を取得
Declare Function Api_GetOEMCP& Lib "kernel32" Alias "GetOEMCP" ()

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

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

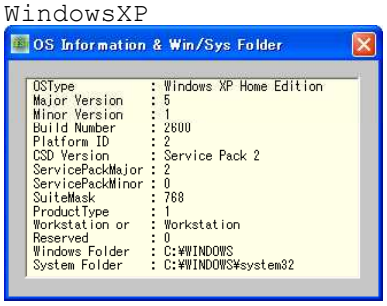
```
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    Ret = Api_GetOEMCP()

    Text2.SetWindowText Str$(Ret)
End Sub

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

OSの情報取得

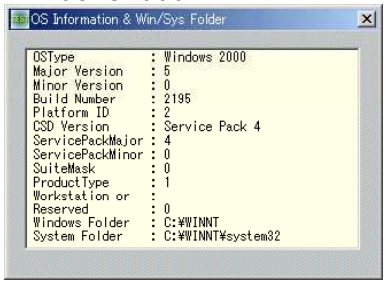
OSの情報を取得します。
GetVersionEx オペレーティングシステムの種類やバージョンに関する情報を取得
GetWindowsDirectory Windowsディレクトリのパス名を取得
GetSystemDirectory Windows のシステムディレクトリのパスを取得

Windows9xとWindows2000以降では分けて取得します。
ついでにWindowsフォル ダ、およびSystemフォルダも取得します。
ビルド番号のデータの格納方法
　全体は、4バイトのLong型変数で、Windows 95 の場合、下位2バイトがビルド番号になっています。
　下位2バイトは、4バイトの値を2の16乗で割った余り、もしくは2の16乗との論理積(AND) の計算から得られます。

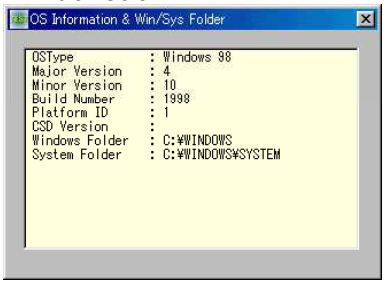
WindowsXP



Windows2000



Windows98




```
'=====================================
'= OSの情報取得
'=    (GetVersionEx.bas)
'=====================================
#include"Windows.bi"

'------------------------------
'OSのバージョン取得(Windows9x)
'------------------------------
Type OSVERSIONINFO
    dwOSVersionInfoSize     As Long         '構造体のバイト数
    dwMajorVersion          As Long         'メジャーバージョン番号
    dwMinorVersion          As Long         'マイナーバージョン番号
    dwBuildNumber           As Long         'ビルド番号
    dwPlatformID            As Long         'プラットフォームのID
    szCSDVersion            As String * 128 'OSに関する付加情報
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)
'------------------------------
'OSのバージョン取得(WindowsNT、2000、XP)
'------------------------------
```

```
Type OSVERSIONINFOEX
    dwOSVersionInfoSize     As Long
    dwMajorVersion          As Long
    dwMinorVersion          As Long
    dwBuildNumber           As Long
    dwPlatformID            As Long
    szCSDVersion            As String * 128'Maintenance string for PSS usage
    wSPMajor                As Integer     'Service Pack Major Version
    wSPMinor                As Integer     'Service Pack Minor Version
    wSuiteMask              As Integer     'Suite Identifier
    bProductType            As Byte        'Server / Workstation / Domain Controller ?
    bReserved               As Byte        'Reserved
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_GetVersionEx2& Lib "Kernel32" Alias "GetVersionExA"
(lpVersionInformation As OSVERSIONINFOEX)

#define VER_NT_SERVER &H3                   '
#define VER_NT_WORKSTATION &H1              '
#define VER_PLATFORM_WIN32_NT 2             'WindowsNT、2000、XP
#define VER_PLATFORM_WIN32_WINDOWS 1        'Windows9x
#define VER_PLATFORM_WIN32s 0               'Windows3.1
#define VER_SERVER_NT &H80000000            '
#define VER_SUITE_BACKOFFICE &H4            '
#define VER_SUITE_BLADE &H400               '
#define VER_SUITE_COMMUNICATIONS &H8        '
#define VER_SUITE_DATACENTER &H80           '
#define VER_SUITE_EMBEDDEDNT &H40           '
#define VER_SUITE_ENTERPRISE &H2            '
#define VER_SUITE_PERSONAL &H200            '
#define VER_SUITE_SINGLEUSERTS &H100        '
#define VER_SUITE_SMALLBUSINESS &H1                 '
#define VER_SUITE_SMALLBUSINESS_RESTRICTED &H20  '
#define VER_SUITE_TERMINAL &H10             '
#define VER_WORKSTATION_NT &H40000000       '

'------------------------------
'Windows•Systemフォルダ取得
'------------------------------
' 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&)

' 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&)

Var Shared Text1 As Object

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

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var ovi As OSVERSIONINFO
    Var OSType As String
    Var MajorVer As Long
    Var MinorVer As Long
    Var PlatformID As Long
    Var CsdVer As String
    Var WorkStr As String
    Var WinPath As String * 255
    Var SysPath As String * 255
    Var nWinPath As String
    Var nSysPath As String
    Var WinPathLen As Long
```

```
Var SysPathLen As Long
Var CrLf As String
Var txt As String
Var Ret As Long

CrLf = Chr$(13, 10)
Text1.SetWindowtext ""

ovi.dwOSVersionInfoSize = Len(ovi)
Ret = Api_GetVersionEx(ovi)
If Ret = 0 Then Exit Sub

'------------------------------
'Windows･Systemフォルダを取得
'------------------------------
WinPathLen = Api_GetWindowsDirectory(WinPath, Len(WinPath))
SysPathLen = Api_GetSystemDirectory(SysPath, Len(SysPath))

nWinPath = Left$(WinPath, InStr(WinPath, Chr$(0)) - 1)
nSysPath = Left$(SysPath, InStr(SysPath, Chr$(0)) - 1)

'------------------------------
'ovi で Version を取得
'------------------------------
MajorVer = ovi.dwMajorVersion
MinorVer = ovi.dwMinorVersion
PlatformID = ovi.dwPlatformID

'------------------------------
'Windows2000 以上と以下で処理を分岐
'------------------------------
Select Case MajorVer
    Case 3
        OSType = "Windows NT 3.51"
    Case 4
        Select Case MinorVer
            Case 0
                If CsdVer = "C" Then
                    OSType = "Windows 95 OSR2"
                Else If PlatformID = VER_PLATFORM_WIN32_NT Then
                    OSType = "Windows NT 4.0"
                Else
                    OSType = "Windows 95"
                End If
            Case 10
                If CsdVer = "A" Then
                    OSType = "Windows 98 SE"
                Else
                    OSType = "Windows 98"
                End If
            Case 90
                OSType = "Windows Me"
            Case Else
                Text1.SetWindowtext " Windows ??"
        End Select

        txt =       " OSType          : " & OSType & CrLf
        txt = txt & " Major Version   : " & Trim$(Str$(ovi.dwMajorVersion)) & CrLf
        txt = txt & " Minor Version   : " & Trim$(Str$(ovi.dwMinorVersion)) & CrLf
        txt = txt & " Build Number    : " & Trim$(Str$(ovi.dwBuildNumber And
                                                  &HFFFF)) & CrLf
        txt = txt & " Platform ID     : " & Trim$(Str$(ovi.dwPlatformID)) & CrLf
        txt = txt & " CSD Version     : " & Left$(ovi.szCSDVersion,
                                 InStr(ovi.szCSDVersion, Che$(0)) - 1) & CrLf
        txt = txt & " Windows Folder  : " & nWinPath & CrLf
        txt = txt & " System Folder   : " & nSysPath
        Text1.SetWindowtext txt
        '------------------------------
        'Windows2000 以上の場合 oviEx で再取得
        '------------------------------
```

```
        Case 5
            Var oviEx As OSVERSIONINFOEX
            Var ProductType As byte
            Var SuiteMask As Integer

            oviEx.dwOSVersionInfoSize = Len(oviEx)

            Ret = Api_GetVersionEx2(oviEx)
            If Ret = 0 Then Exit Sub

            MajorVer = oviEx.dwMajorVersion
            MinorVer = oviEx.dwMinorVersion
            ProductType = oviEx.bProductType
            SuiteMask = oviEx.wSuiteMask

            Select Case MinorVer
                Case 0
                    OSType = "Windows 2000"
                Case 1

                    '------------------------------
                    'Workstation かどうかを取得
                    '------------------------------
                    If ProductType = VER_NT_WORKSTATION Then
                        WorkStr = "Workstation"
                    Else If ProductType = VER_NT_SERVER Then
                        WorkStr = "Server"
                    End If

                    If VER_SUITE_PERSONAL <> 0 Then
                        OSType = "Windows XP Home Edition"
                    Else
                        OSType = "Windows XP Professional Edition"
                    End If
                Case Else
                    Text1.SetWindowtext " Windows ??"
            End Select

            txt =       " OSType          : " & OSType & CrLf
            txt = txt & " Major Version   : " & Trim$(Str$(oviEx.dwMajorVersion)) & CrLf
            txt = txt & " Minor Version   : " & Trim$(Str$(oviEx.dwMinorVersion)) & CrLf
            txt = txt & " Build Number    : " & Trim$(Str$(oviEx.dwBuildNumber And
                                                    &HFFFF)) & CrLf
            txt = txt & " Platform ID     : " & Trim$(Str$(oviEx.dwPlatformID)) & CrLf
            txt = txt & " CSD Version     : " & Left$(oviEx.szCSDVersion,
                                InStr(oviEx.szCSDVersion, Chr$(0)) - 1) & CrLf
            txt = txt & " ServicePackMajor : " & Trim$(Str$(oviEx.wSPMajor)) & CrLf
            txt = txt & " ServicePackMinor : " & Trim$(Str$(oviEx.wSPMinor)) & CrLf
            txt = txt & " SuiteMask       : " & Trim$(Str$(oviEx.wSuiteMask)) & CrLf
            txt = txt & " ProductType     : " & Trim$(Str$(oviEx.bProductType)) & CrLf
            txt = txt & " Workstation or  : " & WorkStr & CrLf
            txt = txt & " Reserved        : " & Trim$(Str$(oviEx.bReserved)) & CrLf
            txt = txt & " Windows Folder  : " & nWinPath & CrLf
            txt = txt & " System Folder   : " & nSysPath

            Text1.SetWindowtext txt
        Case Else
            Text1.SetWindowtext " Windows ??"
    End Select
End Sub

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

osの違いによるフォントサイズ

Virtual PC 上の Windows 2000 で下記コードをコンパイルし、Windows XP 及び Virtual PC 上の Windows 98 で exe を実行してみました。
左側のボタンは、プロパティで設定、右側のボタンはコードで明示的に設定しています。
FixedSys は、18P しか選択できず、コードでサイズを設定しても 18P は変化しないようです。
環境:Dell GX260、Windows XP Pro、Windows 2000 及び Windows 98 は Vietual PC 上で走らせています。
左:Windows XP での実行結果　　中:Windows 2000での実行結果　　右:Windows 98での実行結果


```
'=======================================
'= フォントサイズをプロパティ及びコードで設定した場合の差
'=    (FontSizeOS.bas)
'=======================================
#include "Windows.bi"

Var Shared Button4 As Object
Var Shared Button5 As Object
Var Shared Button6 As Object

Button4.Attach GetDlgItem("Button4") : Button4.SetFontSize 11
Button5.Attach GetDlgItem("Button5") : Button5.SetFontSize 12
Button6.Attach GetDlgItem("Button6") : Button6.SetFontSize 14

While 1
    WaitEvent
Wend
Stop
End
```

osバージョン取得

GetVersionEx OSのバージョンを取得

各OS上での実行結果


```
'=======================================
'= OSのバージョンを取得
'=    (GetVersionEx2.bas)
'=    ※Windows 8 まで
'=======================================
#include "Windows.bi"
```

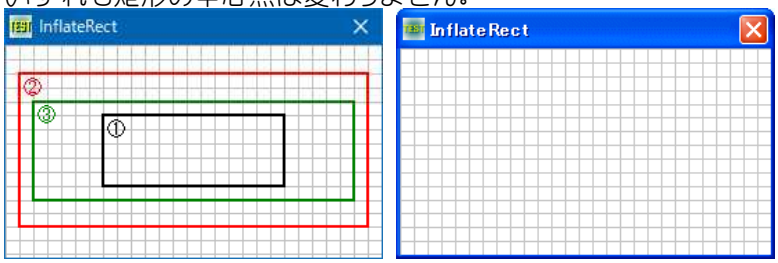
```
Type OSVERSIONINFO
    dwOSVersionInfoSize     As Long
    dwMajorVersion          As Long
    dwMinorVersion          As Long
    dwBuildNumber           As Long
    dwPlatformId            As Long
    szCSDVersion            As String * 128
End Type

#define VER_PLATFORM_WIN32_WINDOWS 1      'Windows9x
#define VER_PLATFORM_WIN32_NT 2           'WindowsNT、2000、XP、Vista、7

' 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 )

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 GetWinPlatform() As String
Function GetWinPlatform() As String
    Var strPlatForm  As String
    Var osvi As OSVERSIONINFO
    Var Ret As Long

    osvi.dwOSVersionInfoSize = Len(osvi)
    Ret = Api_GetVersionEx(osvi)

    strPlatForm = "未知のオペレーティングシステム"

    If osvi.dwPlatformId = VER_PLATFORM_WIN32_WINDOWS Then
        If osvi.dwMinorVersion = 0 Then
            strPlatForm = "Windows 95"
            If osvi.szCSDVersion = "B" Then
                strPlatForm = strPlatForm & " OSR2"
            Else
                strPlatForm = strPlatForm & Left$(osvi.szCSDVersion, 2)
            End If
        Else If osvi.dwMinorVersion = 10 Then
            strPlatForm = "Windows 98"
                If osvi.szCSDVersion = "A" Then
                    strPlatForm = strPlatForm & " SE"
                End If
        Else If osvi.dwMinorVersion = 90 Then
            strPlatForm = "Windows ME"
        Else
            strPlatForm = "Win 32s"
        End If
    Else If osvi.dwPlatformId = VER_PLATFORM_WIN32_NT Then
        If osvi.dwMajorVersion = 4 Then
            strPlatForm = "Windows NT"
        Else If osvi.dwMajorVersion = 5 Then
            If osvi.dwMinorVersion = 0 Then
                strPlatForm = "Windows 2000"
            Else If osvi.dwMinorVersion = 1 Then
                strPlatForm = "Windows XP"
            Else If osvi.dwMinorVersion = 2 Then
                strPlatForm = "Windows 2003"
            End If
        Else If osvi.dwMajorVersion = 6 Then
            If osvi.dwMinorVersion = 0 Then
                strPlatForm = "Windows Vista"
            Else If osvi.dwMinorVersion = 1 Then
                strPlatForm = "Windows 7"
```

```
            Else If osvi.dwMinorVersion = 2 Then
                strPlatForm = "Windows 8"
            ' Else If osvi.dwMinorVersion = 3 Then
            '     strPlatForm = "Windows 8.1"
              End If
          End If
    End If

    GetWinPlatform = strPlatForm
End Function

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Text1.SetWindowText GetWinPlatform
End Sub

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

RECT構造体の座標を拡大･縮小

RECT構造体の座標を拡大･縮小します。
InflateRect RECT構造体の座標を拡大･縮小

①で描画した矩形から横方向に60ドット、縦方向に30ドット拡大描画②、さらに②から横方向に10ドット、縦方向に20ドット縮小描画③します。
いずれも矩形の中心点は変わりません。




```
'=======================================
'= RECT構造体の座標値を拡大･縮小
'=    (InflateRect.bas)
'=======================================
#include "Windows.bi"

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

' RECT構造体の座標値を拡大･縮小
Declare Function Api_InflateRect& Lib "user32" Alias "InflateRect" (lpRect As RECT, ByVal
x&, ByVal y&)

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

```
Sub MainForm_Start()
    Var rct As RECT
    Var Ret As Long

    For x = 0 To 300 Step 10
        Line(x, 0) - (x, 300), , 14
    Next
    For y = 0 To 300 Step 10
        Line(0, y) - (300, y), , 14
    Next

    '①初回指定した四角形
    rct.Left = 70
    rct.Top = 50
    rct.Right = 200
    rct.Bottom = 100

    SetDrawWidth 2

    Line(rct.Left, rct.Top) - (rct.Right, rct.Bottom), , 0, b
    Symbol(rct.Left + 3, rct.Top + 3), "①", 1, 1, 0

    '②X軸60ドット、Y軸30ドット拡大
    Ret = Api_InflateRect(rct, 60, 30)
    Line(rct.Left, rct.Top) - (rct.Right, rct.Bottom), , 5, b
    Symbol(rct.Left + 3, rct.Top + 3), "②", 1, 1, 5

    '③②に対しX軸10ドット、Y軸20ドット縮小
    Ret = Api_InflateRect(rct, -10, -20)
    Line(rct.Left, rct.Top) - (rct.Right, rct.Bottom), , 8, b
    Symbol(rct.Left + 3, rct.Top + 3), "③", 1, 1, 8
End Sub

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

RGBからHLS形式へ変換

ColorRGBToHLS RGBからhue･luminance･saturation(HLS)形式に変換
ColorHLSToRGB hue･luminance･saturation(HLS)からRGB形式に変換


```
'=====================================
'= RGBからHLS形式へ変換
'=    (ColorRGBToHLS.bas)
'=====================================
#include "Windows.bi"

' RGBからhue･luminance･saturation(HLS)形式に変換
Declare Sub Api_ColorRGBToHLS Lib "Shlwapi" Alias "ColorRGBToHLS" (ByVal clrRGB&, pwHue%,
pwLuminance%, pwSaturation%)
```

```
' hue･luminance･saturation(HLS)からRGB形式に変換
Declare Function Api_ColorHLSToRGB& Lib "Shlwapi" Alias "ColorHLSToRGB" (ByVal wHue%,
ByVal wLuminance%, ByVal wSaturation%)

Var Shared Picture1 As Object
Var Shared Picture2 As Object
Var SHared Text(2) As Object
Var Shared Bitmap As Object

Picture1.Attach GetDlgItem("Picture1")
Picture2.Attach GetDlgItem("Picture2")
For i = 0 To 2
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1))) : Text(i).SetFontSize 14
Next i
BitmapObject Bitmap

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Bitmap.LoadFile "flower.bmp"
    Picture1.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteBitmap

    Picture1.SetMousePointer 3
End Sub

'=====================================
'=
'=====================================
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 C As Long
    Var R As Long
    Var G As Long
    Var B As Long
    Var H As Integer                'Hue
    Var L As Integer                'Luminance
    Var S As Integer                'Saturation

    'RGB値を取得
    C = Picture1.GetPixel(X, Y)
    R = (C And &HFF)
    G = (C And &HFF00) / 256
    B = (C And &HFF0000) / 65536

    '取得したRGB値を背景色に
    Picture2.Cls
    Picture2.SetBackColor RGB(R, G, B)

    'HLSに変換
    Api_ColorRGBToHLS RGB(R, G, B), H, L, S

    '値の範囲(0～240)
    Text(0).SetWindowText "色相 : " & Format$(H / 240 * 360, "###") & " °"
    Text(1).SetWindowText "明度 : " & Format$(L / 240 * 100, "###") & " %"
    Text(2).SetWindowText "彩度 : " & Format$(S / 240 * 100, "###") & " %"
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
    Picture1.SetMousePointer 0
End Sub
```

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

TCP統計値を取得

GetTcpStatistics TCP統計値を取得

MIB_TCPSTATUS構造体からTCP(Transmission Control Protocol)統計値を取得します。


```
'=====================================
'= TCP統計値を取得
'=    (GetTcpStatistics.bas)
'=====================================
#include "Windows.bi"

Type MIB_TCPSTATS
    dwRtoAlgorithm    As Long              'Algorithmの種類(MIB_TCP_RTO_･･･参照)
    dwRtoMin          As Long              'RTOの最小値(単位msec)
    dwRtoMax          As Long              'RTOの最大値(単位msec)
    dwMaxConn         As Long              '最大コネクション数(-1は可変長)
    dwActiveOpens     As Long              'コネクションを初期化している状態のコネクション数
    dwPassiveOpens    As Long              'listenしている状態の数
    dwAttemptFails    As Long              'コネクション確立失敗の数
    dwEstabResets     As Long              'Resetされたコネクションの数
    dwCurrEstab       As Long              '現在確立されているコネクション数
    dwInSegs          As Long              '受信したセグメント数
    dwOutSegs         As Long              '送信したセグメント数
    dwRetransSegs     As Long              '再送したセグメント数
    dwInErrs          As Long              '受信したエラーの数
    dwOutRsts         As Long              'RSTフラグが立った状態のセグメントを受け取った数
    dwNumConns        As Long              '現在ある全てのコネクションの数
End Type

#define MIB_TCP_RTO_OTHER 1               'Otherタイムアウトを示す
#define MIB_TCP_RTO_CONSTANT 2            'Constant Timeoutタイムアウト
#define MIB_TCP_RTO_RSRE 3                'MIL-STD-1778 Appendix Bタイムアウト
#define MIB_TCP_RTO_VANJ 4                'Van Jacobson's Algorithmタイムアウト
#define MIB_TCP_MAXCONN_DYNAMIC (-1&)     '接続最大数がダイナミック

' TCP統計値を取得
Declare Function Api_GetTcpStatistics& Lib "iphlpapi" Alias "GetTcpStatistics" (pStats
As MIB_TCPSTATS)

Var Shared List1 As Object
Var Shared Button1 As Object

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

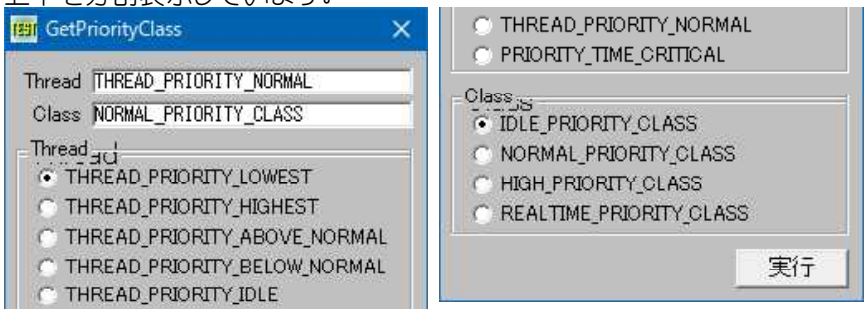
```
'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var mt As MIB_TCPSTATS
    Var Item As String
    Var Ret As Long

    'リストビューを初期化
    List1.Resetcontent

    'TCP統計値を取得
    Ret = Api_GetTcpStatistics(mt)

    'RtoAlgorithmを表示(タイムアウトによって分岐)
    Select Case mt.dwRtoAlgorithm

        'Otherのとき
        Case MIB_TCP_RTO_OTHER
            Item = "OTHER"

        'Constant Timeoutのとき
        Case MIB_TCP_RTO_CONSTANT
            Item = "CONSTANT"

        'MIL-STD-1778 Appendix Bのとき
        Case MIB_TCP_RTO_RSRE
            Item = "RSRE"

        'Van Jacobson's Algorithmのとき
        Case MIB_TCP_RTO_VANJ
            Item = "VANJ"

        'その他のとき
        Case Else
            Item = "UNKNOWN"
    End Select

    List1.AddString "RtoAlgorithm : " & Item

    'RtoMinを表示
    List1.AddString "RtoMin      : " & Trim$(Str$(mt.dwRtoMin))

    'RtoMaxを表示
    List1.AddString "RtoMax      : " & Trim$(Str$(mt.dwRtoMax))

    'MaxConnを表示
    'ダイナミックのときは
    If mt.dwMaxConn = MIB_TCP_MAXCONN_DYNAMIC Then
        Item = "dynamic"

    'ダイナミックではないときは
    Else
        Item = Trim$(Str$(mt.dwMaxConn))
    End If
    List1.AddString "MaxConn     : " & Item

    'ActiveOpensを表示
    Item = Trim$(Str$(mt.dwActiveOpens))
    List1.AddString "ActiveOpens  : " & Item

    'PassiveOpensを表示
    Item = Trim$(Str$(mt.dwPassiveOpens))
    List1.AddString "PassiveOpens : " & Item

    'AttemptFailsを表示
    Item = Trim$(Str$(mt.dwAttemptFails))
    List1.AddString "AttemptFails : " & Item
```

```
    'EstabResetsを表示
    Item = Trim$(Str$(mt.dwEstabResets))
    List1.AddString "EstabResets  : " & Item

    'CurrEstabを表示
    Item = Trim$(Str$(mt.dwCurrEstab))
    List1.AddString "CurrEstab    : " & Item

    'InSegsを表示
    Item = Trim$(Str$(mt.dwInSegs))
    List1.AddString "InSegs       : " & Item

    'OutSegsを表示
    Item = Trim$(Str$(mt.dwOutSegs))
    List1.AddString "OutSegs      : " & Item

    'RetransSegsを表示
    Item = Trim$(Str$(mt.dwRetransSegs))
    List1.AddString "RetransSegs  : " & Item

    'InErrsを表示
    Item = Trim$(Str$(mt.dwInErrs))
    List1.AddString "InErrs       : " & Item

    'OutRstsを表示
    Item = Trim$(Str$(mt.dwOutRsts))
    List1.AddString "OutRsts      : " & Item

    'NumConnsを表示
    Item = Trim$(Str$(mt.dwNumConns))
    List1.AddString "NumConns     : " & Item
End Sub

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

Thread･Classの優先順位設定と取得

GetCurrentThread カレントスレッドの擬似ハンドルを取得
GetCurrentProcess 現在のプロセスに対応する疑似ハンドルを取得
SetThreadPriority 指定したスレッドの優先順位を設定
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
GetPriorityClass 指定のプロセスのプライオリティクラスを取得

上下を分割表示しています。




```
'=====================================
'= Thread･Classの優先順位設定と取得
'=    (SetPriorityClass2.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" ()

' 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" ()

' 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&)

' 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&)

#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_LOWEST -2          '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(THREAD_BASE_PRIORITY_MIN)
#define THREAD_PRIORITY_HIGHEST 2          '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(THREAD_BASE_PRIORITY_MAX)
#define THREAD_PRIORITY_ABOVE_NORMAL 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(THREAD_PRIORITY_HIGHEST-1)
#define THREAD_PRIORITY_BELOW_NORMAL -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(THREAD_PRIORITY_LOWEST+1)
#define THREAD_PRIORITY_IDLE -15           '優先順位クラスが、IDLE_PRIORITY_CLASS、
                                           NORMAL_PRIORITY_CLASS、HIGH_PRIORITY_CLAS
#define THREAD_PRIORITY_NORMAL 0           'スレッド標準の相対優先順位値
#define THREAD_PRIORITY_TIME_CRITICAL 15   '優先順位クラスが、IDLE_PRIORITY_CLASS、
                                           NORMAL_PRIORITY_CLASS、HIGH_PRIORITY_CLAS
#define THREAD_PRIORITY_ERROR_RETURN &H7FFFFFFF

#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 NORMAL_PRIORITY_CLASS &H20         '通常クラス(一般的なプロセス)
#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)

Var Shared Text(3) As Object
Var shared Button1 As Object
Var Shared Group(1) As Object
Var Shared TRadio(6) As Object
Var Shared CRadio(3) As Object

For i = 0 To 6
    If i < 4 Then
        Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1))) : Text(i).SetFontSize 12
        CRadio(i).Attach GetDlgItem("CRadio" & Trim$(Str$(i + 1)))
:CRadio(i).SetFontSize 12
        If i < 2 Then
            Group(i).Attach GetDlgItem("Group" & Trim$(Str$(i + 1))) :
Group(i).SetFontSize 12
        End If
    End If
    TRadio(i).Attach GetDlgItem("TRadio" & Trim$(Str$(i + 1))) : TRadio(i).SetFontSize 12
Next i
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
```

```
Var Shared hThread As Long
Var Shared hProcess As Long

'=====================================
'=
'=====================================
Declare Function optT bdecl () As Integer
Function optT()
    optT = Val(Mid$(GetDlgRadioSelect("TRadio1"), 7)) - 1
End Function

'=====================================
'=
'=====================================
Declare Function optC bdecl () As Integer
Function optC()
    optC = Val(Mid$(GetDlgRadioSelect("CRadio1"), 7)) - 1
End Function

'=====================================
'=
'=====================================
Declare Function GetPriority(strT As String, strC As String) As Integer
Function GetPriority(strT As String, strC As String) As Integer
    Var lThread As Long
    Var lClass As Long

    lThread = Api_GetThreadPriority(hThread)
    lClass = Api_GetPriorityClass(hProcess)

    If lThread = THREAD_PRIORITY_ERROR_RETURN Or lClass = 0 Then
        Exit Function
    End If

    GetPriority = True
    Select Case lClass
        Case IDLE_PRIORITY_CLASS
            strC = "IDLE_PRIORITY_CLASS"
        Case NORMAL_PRIORITY_CLASS
            strC = "NORMAL_PRIORITY_CLASS"
        Case HIGH_PRIORITY_CLASS
            strC = "HIGH_PRIORITY_CLASS"
        Case REALTIME_PRIORITY_CLASS
            strC = "REALTIME_PRIORITY_CLASS"
    End Select

    Select Case lThread
        Case THREAD_PRIORITY_LOWEST
            strT = "LOWEST"
        Case THREAD_PRIORITY_HIGHEST
            strT = "HIGHEST"
        Case THREAD_PRIORITY_ABOVE_NORMAL
            strT = "ABOVE_NORMAL"
        Case THREAD_PRIORITY_BELOW_NORMAL
            strT = "BELOW_NORMAL"
        Case THREAD_PRIORITY_IDLE
            strT = "IDLE"
        Case THREAD_PRIORITY_NORMAL
            strT = "NORMAL"
        Case THREAD_PRIORITY_TIME_CRITICAL
            strT = "TIME_CRITICAL"
    End Select

    strT = "THREAD_PRIORITY_" & strT
End Function

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

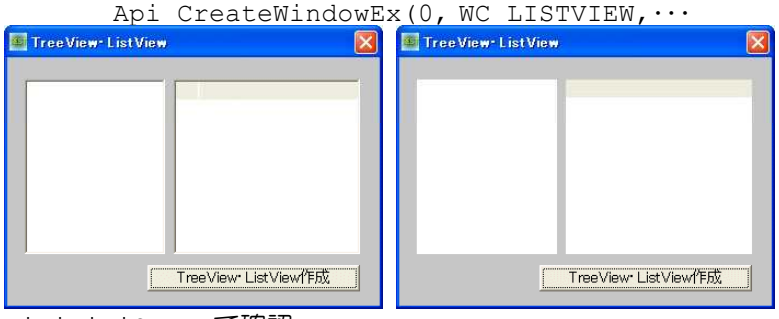
```
Sub MainForm_Start()
    Var Thread As String
    Var Class As String
    Var Ret As Long

    hThread = Api_GetCurrentThread()
    hProcess = Api_GetCurrentProcess()

    If GetPriority(Thread, Class) = True Then
        Text(0).SetWindowText Thread
        Text(1).SetWindowText Class

        Ret = Api_SetThreadPriority(hThread, THREAD_PRIORITY_NORMAL)
        Ret = Api_SetPriorityClass(hProcess, NORMAL_PRIORITY_CLASS)
    Else
        A% = MessageBox("", "失敗しました！", 0, 2)
        End
    End If
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var lT As Long
    Var lC As Long
    Var Thread As String
    Var Class As String
    Var Ret As Long

    Select Case optT
        Case 0
            lT = THREAD_PRIORITY_LOWEST
        Case 1
            lT = THREAD_PRIORITY_HIGHEST
        Case 2
            lT = THREAD_PRIORITY_ABOVE_NORMAL
        Case 3
            lT = THREAD_PRIORITY_BELOW_NORMAL
        Case 4
            lT = THREAD_PRIORITY_IDLE
        Case 5
            lT = THREAD_PRIORITY_NORMAL
        Case 6
            lT = THREAD_PRIORITY_TIME_CRITICAL
    End Select

    Select Case optC
        Case 0
            lC = IDLE_PRIORITY_CLASS
        Case 1
            lC = NORMAL_PRIORITY_CLASS
        Case 2
            lC = HIGH_PRIORITY_CLASS
        Case 3
            lC = REALTIME_PRIORITY_CLASS
    End Select

    Ret = Api_SetThreadPriority(hThread, lT)
    Ret = Api_SetPriorityClass(hProcess, lC)

    If GetPriority(Thread, Class) = True Then
        Text(0).SetWindowText Thread
        Text(1).SetWindowText Class
    Else
        A% = MessageBox("", "失敗しました！", 0, 2)
        End
    End If
End Sub
```

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

TreeView･ListViewの作成

TreeView、ListViewを作成します。(フレームの作成のみ=未完成)
InitCommonControls 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
CreateWindowEx ウインドウ(コントロール)を作成
DestroyWindow CreateWindowExの解放
SendMessage ウィンドウにメッセージを送信

左:Api_CreateWindowEx(WS_EX_CLIENTEDGE, WC_TREEVIEW,･･･
　　Api_CreateWindowEx(WS_EX_CLIENTEDGE, WC_LISTVIEW,･･･
右:Api_CreateWindowEx(0, WC_TREEVIEW,･･･
　　Api_CreateWindowEx(0, WC_LISTVIEW,･･･



MiniMiniSpy--で確認


```
'====================================
'= TreeView･ListViewの作成
'=    (TreeListView.bas)
'====================================
#include "Windows.bi"

#define WS_VISIBLE &H10000000                '表示する
#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_EX_CLIENTEDGE &H200               '縁が沈んで見える境界線を持つウィンドウを指定
#define WC_LISTVIEW "SysListView32"
#define WC_TREEVIEW "SysTreeView32"

#define ICC_LISTVIEW_CLASSES &H1             'リストビュー、ヘッダーコントロール
#define ICC_TREEVIEW_CLASSES &H2             'ツリービュー、ツールチップ
#define ICC_WIN95_CLASSES &HFF               'すべてのコントロール

#define LVM_DELETEALLITEMS &H1009            'すべてのアイテムを削除
#define LVM_DELETECOLUMN &H101C              'カラムを削除
#define LVM_DELETEITEM &H1008                'アイテムを削除
#define LVM_FINDITEM &H100D                  'アイテムを検索
#define LVM_GETBKCOLOR &H1000                '背景色を取得
#define LVM_GETEXTENDEDLISTVIEWSTYLE &H1037  '拡張スタイルを取得
#define LVM_GETHEADER &H101F                 'ヘッダコントロールを取得
#define LVM_GETITEM &H1005                   'アイテムの属性を取得
#define LVM_GETITEMCOUNT &H1004              'アイテムの数を取得
```

```
#define LVM_GETNextITEM &H100C              'Ｉ定した属性を持つアイテムを取得
#define LVM_GETTEXTCOLOR &H1023             'テキストの文字色を取得
#define LVM_GETTEXTBKCOLOR &H1025           'テキストの背景色を取得
#define LVM_INSERTCOLUMN &H101B             '新しいカラム(列)を挿入
#define LVM_INSERTITEM &H1007               '新しいアイテムを挿入
#define LVM_SETBKCOLOR &H1001               '背景色の設定
#define LVM_SETEXTENDEDLISTVIEWSTYLE &H1036 '拡張スタイルの設定
#define LVM_SETIMAGELIST &H1003             'イメージリストの割り当て
#define LVM_SETITEM &H1006                  'アイテム・サブアイテムの属性を設定・変更
#define LVM_SETTEXTBKCOLOR &H1026           'テキストの背景色を設定
#define LVM_SETTEXTCOLOR &H1024             'テキストの文字色を設定
#define LVS_TypeMASK &H3
#define LVS_REPORT &H1                      '詳細表示。最初の列が常に左寄せにされる

#define TVS_CHECKBOXES &H100                'アイテムにチェックボックスを付ける。いったんこのスタイルが設定されると、チェックボックスを取り除くこ
#define TVS_DISABLEDRAGDROP &H10            'ツリービューが親ウィンドウにTVN_BEGINDRAG 通知メッセージを送らないようにする
#define TVS_EDITLABELS &H8                  'ユーザーがツリービューのアイテムのテキストを編集できるようにする
#define TVS_FULLROWSELECT &H1000            'アイテムをその列全体で選択できるようにする。アイテムのある列のどの部分をクリックしてもそのアイテムが
#define TVS_HASBUTTONS &H1                  '子アイテムを持つときに親アイテムの横に+や-のボタンを表示
#define TVS_HASLINES &H2                    'アイテムを線でつなぐ
#define TVS_INFOTIP &H800                   'ツリービューはツールチップ情報を得るために親ウィンドウにTVN_GETINFOTIP通知メッセージを送る。
#define TVS_LINESATROOT &H4                 '一番上のアイテムに線を付ける。TVS_HASLINESスタイルが指定されていない場合は無視される
#define TVS_NONEVENHEIGHT &H4000            'ツリービューにTVM_SETITEMHEIGHTメッセージを送ることで、アイテムの高さを設定できるようにする(デフォル
#define TVS_NOSCROLL &H2000                 'ツリービューがスクロールしないようにする。スクロールバーも表示しない
#define TVS_NOTOOLTIPS &H80                 'アイテムがツリービューからはみ出ているときにツールチップが付かないようにする(version4.70以降)
#define TVS_RTLREADING &H40                 'アラビア語やヘブライ語などのシステムにおいて、右から左向きに表示。日本語のシステムでは指定できない(v
#define TVS_SHOWSELALWAYS &H20              'ツリービューがフォーカスを持っていない状態でも、アイテム選択状態が表示されるようにする
#define TVS_SINGLEEXPAND &H400              '選択されたアイテムのみが展開されるようにする(version4.70以降)
#define TVS_TRACKSELECT &H200               'マウスカーソルがアイテムの上に来たときに下線が付く(version4.70以降)
' 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" ()

' ウインドウ(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&)

' 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。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&)

Var Shared hTreeViewCtrl As Long
Var Shared hListViewCtrl As Long

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

```
    '初期化
    Api_InitCommonControls

    'TreeView作成
    hTreeViewCtrl = Api_CreateWindowEx(WS_EX_CLIENTEDGE, WC_TREEVIEW, "TreeView1",
WS_CHILD Or WS_VISIBLE Or TVS_HASBUTTONS Or TVS_HASLINES, 16, 20, 128, 160, GethWnd, 0, 0,
0)

    'ListView作成
    hListViewCtrl = Api_CreateWindowEx(WS_EX_CLIENTEDGE, WC_LISTVIEW, "ListView1",
WS_CHILD Or WS_VISIBLE Or LVS_REPORT, 152, 20, 170, 160, GethWnd, 0, 0, 0)
End Sub

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

    Ret = Api_DestroyWindow(hTreeViewCtrl)
    Ret = Api_DestroyWindow(hListViewCtrl)
End Sub

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

TWAIN機器からの入力

TWAIN対応機器からの入力と、その画像を描画します。
EZTW32.DLLというフリーソフトを見つけました。
http://www.geocities.com/smigman.geo/mci/home.html
http://www.nodevice.com/dll/EZTW32_DLL/item6371.html
イメージスキャナ、Webカメラなどを操作できそうです。
TWAIN_SelectImageSource TWAIN機器の選択
TWAIN_AcquireToClipboard TWAIN機器から取得したデータをクリップボードへ転送
TWAIN_AcquireToFilename TWAIN機器から取得したデータをファイル保存

「TWAIN選択」:TWAIN対応機器を選択します。
「Image取得」:読み取ったデータをクリップボードに転送します。
「取得保存」:読み取ったデータをBitmap型式ファイルで保存します。
「描画」:一旦保存したファイルを読み出して描画させています。


```
'=======================================
'= TWAIN機器からの入力
'=    (Twain.bas)
'=======================================
#include "Windows.bi"
```

```
' TWAIN機器の選択
Declare Function TWAIN_SelectImageSource& Lib "Eztw32" Alias "TWAIN_SelectImageSource"
(ByVal hWnd&)

' TWAIN機器から取得したデータをクリップボードへ転送
Declare Function TWAIN_AcquireToClipboard& Lib "Eztw32" Alias
"TWAIN_AcquireToClipboard" (ByVal hwndApp&, ByVal wPixTypes&)

' TWAIN機器から取得したデータをファイル保存
Declare Function TWAIN_AcquireToFilename& Lib "Eztw32" Alias "TWAIN_AcquireToFilename"
(ByVal hwndApp&, ByVal sFile$)

#define CF_BITMAP 2                    'ビットマップのデータ(HBITMAP)
#define CF_DIB 8                       '構造体とビットマップビットからなるメモリオブジェクト
#define CF_METAFILEPICT 3              'メタファイル画像形式。METAFILEPICT構造体のメモリオブ
                                       ジェクト
#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

Var Shared Picture1 As Object
Var Shared Edit1 As Object
Var Shared Button(3) As Object
Var Shared Radio1 As Object
Var Shared Radio2 As Object
Var Shared Bitmap1 As Object
BitmapObject Bitmap1

Picture1.Attach GetDlgItem("Picture1")
Edit1.Attach getDlgItem("Edit1") : Edit1.SetFontSize 14
Radio1.Attach getDlgItem("Radio1") : Radio1.SetFontSize 14
Radio2.Attach getDlgItem("Radio2") : Radio2.SetFontSize 14

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

Var Shared FileName As String

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

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    ShowWindow -1
End Sub

'=====================================
'= TWAIN機器の選択
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    Ret = TWAIN_SelectImageSource(GethWnd)
End Sub

'=====================================
'= スキャンしクリップボードへ転送
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
```

```
    Var Ret As Long

    Picture1.Cls
    ClearCB

    Ret = TWAIN_AcquireToClipboard(GethWnd, 0)

    If Ret = 0 Then
        A% = MessageBox(GetWindowText, "イメージの取得･クリップボードへの転送は失敗しました！", 0,
2)
        Exit Sub
    End If
End Sub

'=====================================
'= スキャンし取得データをファイルに保存
'=====================================
Declare Sub Button3_on edecl ()
Sub Button3_on()
    Var Ret As Long

    FileName = Edit1.GetWindowText
    Ret = TWAIN_AcquireToFilename(GethWnd, FileName)
End Sub

'=====================================
'= ファイルを読み込みピクチャボックスに描画
'=====================================
Declare Sub Button4_on edecl ()
Sub Button4_on()
    Picture1.Cls

    On Error Goto *ErrTrap

    Bitmap1.LoadFile Edit1.GetWindowText

    If Flg = 0 Then
        Picture1.DrawBitmap Bitmap1, 0, 0
    Else
        Picture1.StretchBitmap Bitmap1, 0, 0, Picture1.GetWidth, Picture1.GetHeight
    End If

    Bitmap1.DeleteBitmap

*ErrRet
    On Error Goto 0
    Exit Sub

*ErrTrap
    A% = MessageBox(GetWindowText, "ファイルが見あたりません！", 0, 2)
    Resume *ErrRet
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Resize edecl ()
Sub MainForm_Resize()
    If GetWidth < 350 Or GetHeight < 354 Then
        SetWindowSize 350, 354
    End If

    Picture1.SetWindowSize GetWidth - 30, GetHeight - 110
    Edit1.MoveWindow 176, GetHeight - 96
    Radio1.MoveWindow 6, GetHeight - 94
    Radio2.MoveWindow 94, GetHeight - 94
    Button(0).MoveWindow 12, GetHeight - 68
    Button(1).MoveWindow 92, GetHeight - 68
    Button(2).MoveWindow 172, GetHeight - 68
    Button(3).MoveWindow 252, GetHeight - 68
```

```
End Sub

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

参考

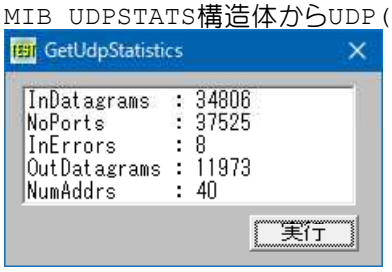
```
Declare Function TWAIN_AbortAllPendingXfers& Lib "Eztw32" Alias
"TWAIN_AbortAllPendingXfers" ()
Declare Function TWAIN_AcquireNative& Lib "Eztw32" Alias "TWAIN_AcquireNative" (ByVal
hwndApp&, ByVal wPixTypes&)
Declare Function TWAIN_AcquireToClipboard& Lib "Eztw32" Alias
"TWAIN_AcquireToClipboard" (ByVal hwndApp&, ByVal wPixTypes&)
Declare Function TWAIN_AcquireToFilename& Lib "Eztw32" Alias "TWAIN_AcquireToFilename"
(ByVal hwndApp&, ByVal sFile$)
Declare Function TWAIN_CloseSource& Lib "Eztw32" Alias "TWAIN_CloseSource" ()
Declare Function TWAIN_CloseSourceManager& Lib "Eztw32" Alias
"TWAIN_CloseSourceManager" (ByVal hWnd&)
Declare Function TWAIN_CreateDibPalette& Lib "Eztw32" Alias "TWAIN_CreateDibPalette"
(ByVal hDib&)
Declare Function TWAIN_DibDepth& Lib "Eztw32" Alias "TWAIN_DibDepth" (ByVal hDib&)
Declare Function TWAIN_DibHeight& Lib "Eztw32" Alias "TWAIN_DibHeight" (ByVal hDib&)
Declare Function TWAIN_DibNumColors& Lib "Eztw32" Alias "TWAIN_DibNumColors" (ByVal
hDib&)
Declare Function TWAIN_DibWidth& Lib "Eztw32" Alias "TWAIN_DibWidth" (ByVal hDib&)
Declare Function TWAIN_DisableSource& Lib "Eztw32" Alias "TWAIN_DisableSource" ()
Declare Function TWAIN_EasyVersion& Lib "Eztw32" Alias "TWAIN_EasyVersion" ()
Declare Function TWAIN_EnableSource& Lib "Eztw32" Alias "TWAIN_EnableSource" (ByVal
hWnd&)
Declare Function TWAIN_EndXfer& Lib "Eztw32" Alias "TWAIN_EndXfer" ()
Declare Function TWAIN_GetBitDepth& Lib "Eztw32" Alias "TWAIN_GetBitDepth" ()
Declare Function TWAIN_GetConditionCode& Lib "Eztw32" Alias "TWAIN_GetConditionCode" ()
Declare Function TWAIN_GetCurrentResolution# Lib "Eztw32" Alias
"TWAIN_GetCurrentResolution" ()
Declare Function TWAIN_GetCurrentUnits& Lib "Eztw32" Alias "TWAIN_GetCurrentUnits" ()
Declare Function TWAIN_GetHideUI& Lib "Eztw32" Alias "TWAIN_GetHideUI" ()
Declare Function TWAIN_GetPixelType& Lib "Eztw32" Alias "TWAIN_GetPixelType" ()
Declare Function TWAIN_GetResultCode& Lib "Eztw32" Alias "TWAIN_GetResultCode" ()
Declare Function TWAIN_IsAvailable& Lib "Eztw32" Alias "TWAIN_IsAvailable" ()
Declare Function TWAIN_LoadNativeFromFile& Lib "Eztw32" Alias
"TWAIN_LoadNativeFromFile" (ByVal hFile&)
Declare Function TWAIN_LoadNativeFromFilename& Lib "Eztw32" Alias
"TWAIN_LoadNativeFromFilename" (ByVal sFile$)
Declare Function TWAIN_LoadSourceManager& Lib "Eztw32" Alias "TWAIN_LoadSourceManager"
()
Declare Function TWAIN_MessageHook& Lib "Eztw32" Alias "TWAIN_MessageHook" (lpMsg As Any)
Declare Function TWAIN_NegotiatePixelTypes& Lib "Eztw32" Alias
"TWAIN_NegotiatePixelTypes" (ByVal wPixTypes&)
Declare Function TWAIN_NegotiateXferCount& Lib "Eztw32" Alias
"TWAIN_NegotiateXferCount" (ByVal nXfers&)
Declare Function TWAIN_OpenDefaultSource& Lib "Eztw32" Alias "TWAIN_OpenDefaultSource"
()
Declare Function TWAIN_OpenSourceManager& Lib "Eztw32" Alias "TWAIN_OpenSourceManager"
(ByVal hWnd&)
Declare Function TWAIN_RowSize& Lib "Eztw32" Alias "TWAIN_RowSize" (ByVal hDib&)
Declare Function TWAIN_SelectImageSource& Lib "Eztw32" Alias "TWAIN_SelectImageSource"
(ByVal hWnd&)
Declare Function TWAIN_SetBitDepth& Lib "Eztw32" Alias "TWAIN_SetBitDepth" (ByVal
nBits&)
Declare Function TWAIN_SetBrightness& Lib "Eztw32" Alias "TWAIN_SetBrightness" (ByVal
dBri#)
Declare Function TWAIN_SetContrast& Lib "Eztw32" Alias "TWAIN_SetContrast" (ByVal dCon#)
Declare Function TWAIN_SetCurrentPixelType& Lib "Eztw32" Alias
"TWAIN_SetCurrentPixelType" (ByVal nPixType&)
Declare Function TWAIN_SetCurrentResolution& Lib "Eztw32" Alias
```

```
"TWAIN_SetCurrentResolution" (ByVal dRes#)
Declare Function TWAIN_SetCurrentUnits& Lib "Eztw32" Alias "TWAIN_SetCurrentUnits"
(ByVal nUnits&)
Declare Function TWAIN_State& Lib "Eztw32" Alias "TWAIN_State" ()
Declare Function TWAIN_UnloadSourceManager& Lib "Eztw32" Alias
"TWAIN_UnloadSourceManager" ()
Declare Function TWAIN_WaitForNativeXfer& Lib "Eztw32" Alias "TWAIN_WaitForNativeXfer"
(ByVal hWnd&)
Declare Function TWAIN_WriteDibToFile& Lib "Eztw32" Alias "TWAIN_WriteDibToFile" (ByVal
lpDib&, ByVal hFile&)
Declare Function TWAIN_WriteNativeToFile& Lib "Eztw32" Alias "TWAIN_WriteNativeToFile"
(ByVal hDib&, ByVal hFile&)
Declare Function TWAIN_WriteNativeToFilename& Lib "Eztw32" Alias
"TWAIN_WriteNativeToFilename" (ByVal hDib&, ByVal sFile$)
Declare Sub TWAIN_DrawDibToDC Lib "Eztw32" Alias "TWAIN_DrawDibToDC" (ByVal hDC&, ByVal
nDestX&, ByVal nDestX&, ByVal nWidth&, ByVal nHeight&, ByVal hDib&, ByVal nSrcX&, ByVal
nSrcX&)
Declare Sub TWAIN_ErrorBox Lib "Eztw32" Alias "TWAIN_ErrorBox" (ByVal sMsg$)
Declare Sub TWAIN_FreeNative Lib "Eztw32" Alias "TWAIN_FreeNative" (ByVal hDib&)
Declare Sub TWAIN_ModalEventLoop Lib "Eztw32" Alias "TWAIN_ModalEventLoop" ()
Declare Sub TWAIN_ReadRow Lib "Eztw32" Alias "TWAIN_ReadRow" (ByVal hDib&, ByVal nRow&,
prow As Any)
Declare Sub TWAIN_RegisterApp Lib "Eztw32" Alias "TWAIN_RegisterApp" (ByVal nMajorNum&,
ByVal nMinorNum&, ByVal nLanguage&, ByVal nCountry&, ByVal sVersion$, ByVal sMfg$, ByVal
sFamily$, ByVal sProduct$)
Declare Sub TWAIN_ReportLastError Lib "Eztw32" Alias "TWAIN_ReportLastError" (ByVal
sMsg$)
Declare Sub TWAIN_SetHideUI Lib "Eztw32" Alias "TWAIN_SetHideUI" (ByVal nHide&)
```

UDP統計値を取得

GetUdpStatistics UDP統計値を取得

MIB_UDPSTATS構造体からUDP(User Datagram Protocol)統計値を取得します。




```
'=======================================
'= UDP統計値を取得
'=    (GetUdpStatistics.bas)
'=======================================
#include "Windows.bi"

Type MIB_UDPSTATS
    dwInDatagrams     As Long              '受信したデータグラム数
    dwNoPorts         As Long              'サービスを提供していないポート宛に送られてきて破棄さ
                                           れたデータグラム数
    dwInErrors        As Long              '受信エラー数
    dwOutDatagrams    As Long              '送信したデータグラム数
    dwNumAddrs        As Long              'UDP listener tableのエントリ数
End Type

' UDP統計値を取得
Declare Function Api_GetUdpStatistics& Lib "iphlpapi" Alias "GetUdpStatistics" (pStats
As MIB_UDPSTATS)

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 us As MIB_UDPSTATS
    Var Item As String
    Var Ret As Long

    'リストビューを初期化
    List1.Resetcontent

    'TCP統計値を取得
    Ret = Api_GetUdpStatistics(us)

    'InDatagramsを表示
    Item = Trim$(Str$(us.dwInDatagrams))
    List1.AddString "InDatagrams  : " & Item

    'NoPortsを表示
    Item = Trim$(Str$(us.dwNoPorts))
    List1.AddString "NoPorts      : " & Item

    'InErrorsを表示
    Item = Trim$(Str$(us.dwInErrors))
    List1.AddString "InErrors     : " & Item

    'OutDatagramsを表示
    Item = Trim$(Str$(us.dwOutDatagrams))
    List1.AddString "OutDatagrams : " & Item

    'NumAddrsを表示
    Item = Trim$(Str$(us.dwNumAddrs))
    List1.AddString "NumAddrs     : " & Item
End Sub

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

UNCパスを取得

WNetOpenEnum ネットワークリソースの列挙、または、現在の接続の列挙を開始
WNetEnumResource WNetOpenEnum 関数を呼び出して開始したネットワークリソースの列挙を継続
WNetCloseEnum WNetOpenEnum 関数を呼び出して開始したネットワークリソースの列挙を終了
lstrlen 指定された文字列のバイトまたは文字の長さを返す
lstrcpy 文字列をコピーする

UNC (Universal Naming Convention)
下図の様なLAN接続での例

参考URL
http://www.atmarkit.co.jp/icd/root/91/5787091.html

```
'=====================================
'= UNCパスを取得
'=    (WNetEnumResource.bas)
'=====================================
#include "Windows.bi"

#define RESOURCETYPE_ANY &H0
#define RESOURCE_CONNECTED &H1

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

' ネットワークリソースの列挙、または、現在の接続の列挙を開始
Declare Function Api_WNetOpenEnum& Lib "mpr" Alias "WNetOpenEnumA" (ByVal dwScope&,
ByVal dwType&, ByVal dwUsage&, lpNetResource As NETRESOURCE, lphEnum&)

' WNetOpenEnum 関数を呼び出して開始したネットワークリソースの列挙を継続
Declare Function Api_WNetEnumResource& Lib "mpr" Alias "WNetEnumResourceA" (ByVal
hEnum&, lpcCount&, lpBuffer As Any, lpBufferSize&)

' WNetOpenEnum 関数を呼び出して開始したネットワークリソースの列挙を終了
Declare Function Api_WNetCloseEnum& Lib "mpr" Alias "WNetCloseEnum" (ByVal hEnum&)

' 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 As Any)

' 文字列をコピーする
Declare Function Api_lstrcpy& Lib "Kernel32" Alias "lstrcpyA" (ByVal lpszString1 As Any,
ByVal lpszString2 As Any)

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 Function LetterToUNC(DriveLetter As String) As String
Function LetterToUNC(DriveLetter As String) As String
    Var hEnum As Long
    Var ns(1023) As NETRESOURCE
    Var entries As Long
```

```
    Var nStatus As Long
    Var LocalName As String
    Var UNCName As String
    Var i As Long
    Var r As Long

    '列挙開始
    nStatus = Api_WNetOpenEnum(RESOURCE_CONNECTED, RESOURCETYPE_ANY, 0, ByVal 0, hEnum)

    LetterToUNC = DriveLetter

    '列挙からの成功をチェック
    If ((nStatus = 0) And (hEnum <> 0)) Then

        'エントリー数を設定
        entries = 1024

        'リソースを列挙
        nStatus = Api_WNetEnumResource(hEnum, entries, ns(0), CLng(Len(ns(0))) * 1024)

        '成功をチェック
        If nStatus = 0 Then
            For i = 0 To entries - 1

                'ローカル名を取得
                LocalName = ""
                If ns(i).lpLocalName <> 0 Then
                    LocalName = Space$(Api_lstrlen(ns(i).lpLocalName) + 1)
                    r = Api_lstrcpy(LocalName, ns(i).lpLocalName)
                End If

                '最後のNullを削除
                If Len(LocalName) <> 0 Then
                    LocalName = Left$(LocalName, (Len(LocalName) - 1))
                End If

                If UCase$(LocalName) = UCase$(DriveLetter) Then

                    'リモート名を取得
                    UNCName = ""
                    If ns(i).lpRemoteName <> 0 Then
                        UNCName = Space$(Api_lstrlen(ns(i).lpRemoteName) + 1)
                        r = Api_lstrcpy(UNCName, ns(i).lpRemoteName)
                    End If

                    '最後のNullを削除
                    If Len(UNCName) <> 0 Then
                        UNCName = Left$(UNCName, (Len(UNCName) - 1))
                    End If

                    'UNC pathを返す
                    LetterToUNC = UNCName

                    'loopを抜ける
                    Exit For
                End If
            Next i
        End If
    End If

    ' End enumeration
    nStatus = Api_WNetCloseEnum(hEnum)
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var DriveName As String
```

```
    DriveName = Edit1.GetWindowText
    If Right$(DriveName, 1) <> ":" Then DriveName = DriveName & ":"

    Text1.SetWindowText "UNC path: " & LetterToUNC(DriveName)
End Sub

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

Unicode文字列をANSI文字列に変換

Unicode文字列をANSI文字列に変換します。
GetTimeZoneInformation 現在のタイムゾーンのパラメータを取得
WideCharToMultiByte Unicode文字列をANSI文字列に変換

例では直接取得したStandardName・DaylightNameと、それらの変換後を表示しています。

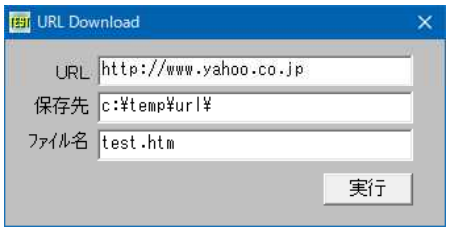

```
'=======================================
'= Unicode文字列をANSI文字列に変換
'=    (WideCharToMultiByte.bas)
'=======================================
#include "Windows.bi"

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

Type TIME_ZONE_INFORMATION
    Bias                As Long
    StandardName        As String * 64
    StandardDate        As SYSTEMTIME
    StandardBias        As Long
    DaylightName        As String * 64
    DaylightDate        As SYSTEMTIME
    DaylightBias        As Long
End Type

' 現在のタイムゾーンのパラメータを取得
Declare Function Api_GetTimeZoneInformation& Lib "kernel32" Alias
"GetTimeZoneInformation" (lpTimeZoneInformation As TIME_ZONE_INFORMATION)

' Unicode文字列をANSI文字列に変換
Declare Function Api_WideCharToMultiByte& Lib "kernel32" Alias "WideCharToMultiByte"
(ByVal CodePage&, ByVal dwFlags&, ByVal lpWideCharStr$, ByVal cchWideChar&, ByVal
lpMultiByteStr$, ByVal cchMultiByte&, ByVal lpDefaultChar$, ByVal lpUsedDefaultChar&)
```

```
Var tzi As TIME_ZONE_INFORMATION
Var TmpOut As String * 32
Var Ret As Long

'タイムゾーン情報取得
Ret = Api_GetTimeZoneInformation(tzi)

Print "変換前"
Print tab(4) & "タイムゾーン:" & left$(tzi.StandardName, instr(tzi.StandardName, Chr$(0))
- 1)
Print tab(4) & "サマータイム:" & left$(tzi.DaylightName, instr(tzi.DaylightName, Chr$(0))
- 1)

Print "変換後"

'タイムゾーンをUnicodeからANSIに変換
Ret = Api_WideCharToMultiByte(0, 0, tzi.StandardName, 64, TmpOut, 32, "", 0)
Print tab(4) & "タイムゾーン:" &  left$(TmpOut, instr(1, TmpOut, Chr$(0)) - 1)

'サマータイムをUnicodeからANSIに変換
Ret = Api_WideCharToMultiByte(0, 0, tzi.DaylightName, 64, TmpOut, 32, "", 0)
Print tab(4) & "サマータイム:" &  left$(TmpOut, instr(1, TmpOut, Chr$(0)) - 1)

Stop
End
```

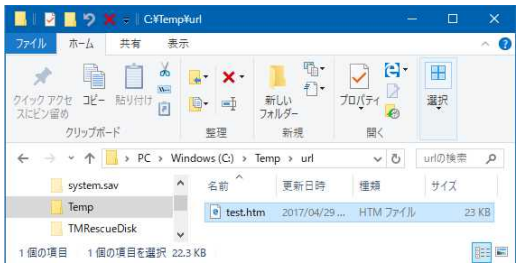
URLダウンロード(Ⅰ)

指定したURLのHTML(PDF)をダウンロードしファイルに保存します。
URLDownloadToFile URLをダウンロードしファイルに保存
MakeSureDirectoryPathExists 新規ディレクトリの作成

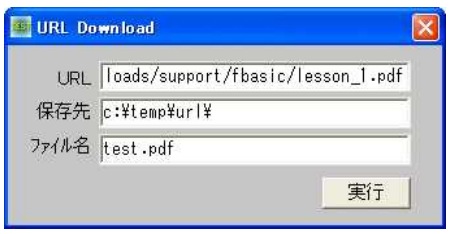
URLDownloadToFileは、フォルダが存在していなければならないようですので
MakeSureDirectoryPathExistsでフォルダを作成しています。

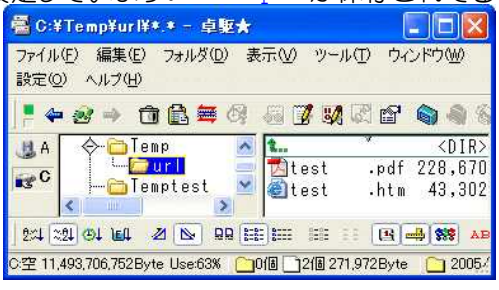


左記ファイルを開いてみます。



URLに"http://jp.fujitsu.com/group/pst/downloads/support/fbasic/lesson_1.pdf"と入力し、ファイル名を「test.pdf」としてみました。
URL入力部は水平スクロールをありに設定しています。test.pdfが保存されてるのが確認できます。





上記ファイルを開いてみます。

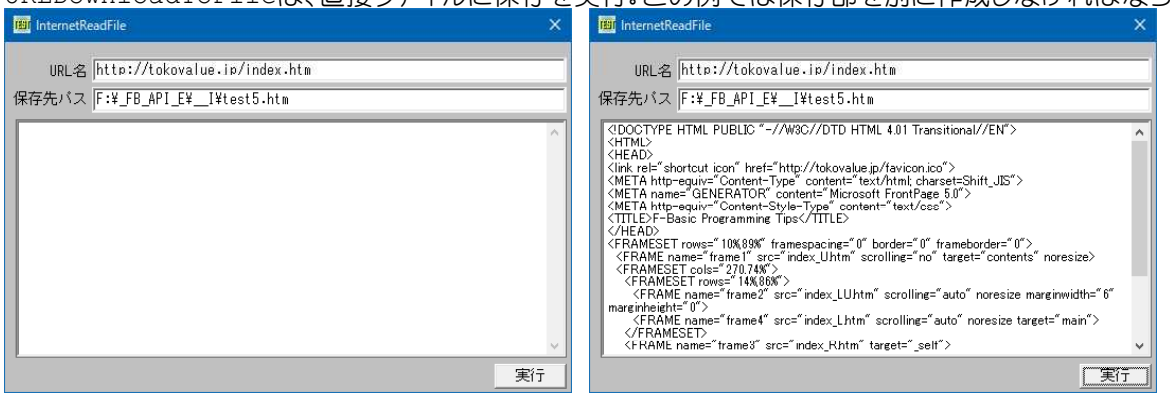

```
'=====================================
'= URLダウンロード
'=    (UrlDownload.bas)
'=====================================
#include "Windows.bi"

' URLのHTML等をダウンロードしファイルに保存
Declare Function Api_URLDownloadToFile& Lib "urlmon" Alias "URLDownloadToFileA" (ByVal
pCaller&, ByVal szURL$, ByVal szFileName$, ByVal dwReserved&, ByVal lpfnCB&)

' 階層化されたフォルダを一気に作成
Declare Function Api_MakeSureDirectoryPathExists& Lib "imagehlp" Alias
"MakeSureDirectoryPathExists" (ByVal lpPath$)

Var Shared Edit(2) As Object
For i = 0 To 2
    Edit(i).Attach GetDlgItem("Edit" & Trim$(Str$(i+1)))
    Edit(i).SetFontSize 14
Next

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    '"http://jp.fujitsu.com/group/pst/downloads/support/fbasic/lesson_1.pdf"
    Edit(0).SetWindowText "http://www.yahoo.co.jp"
    Edit(1).SetWindowText "c:¥temp¥url¥"
    Edit(2).SetWindowText "test.htm"
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Url As String
    Var DirPath As String
    Var sName As String
    Var sFileName As String
    Var Ret As Long

    Url = GetDlgItemText("Edit1")
    DirPath = GetDlgItemText("Edit2")
    sName = GetDlgItemText("Edit3")
    sFileName = DirPath & sName

    SetMousePointer 2
    Ret = Api_MakeSureDirectoryPathExists(DirPath)
    Ret = Api_URLDownloadToFile(0, Url, sFileName, 0, 0)
    SetMousePointer 0
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
```

```
Stop
End
```

URLダウンロード(II)

URLをダウンロードします。
InternetOpen インターネットのハンドルの作成
InternetCloseHandle Win32インターネット関数のハンドルのクローズ
InternetReadFile インターネット上のファイルの読み込み
InternetOpenUrl URLのオープン

URLDownloadToFileは、直接ファイルに保存を実行。この例では保存部を別に作成しなければならない。

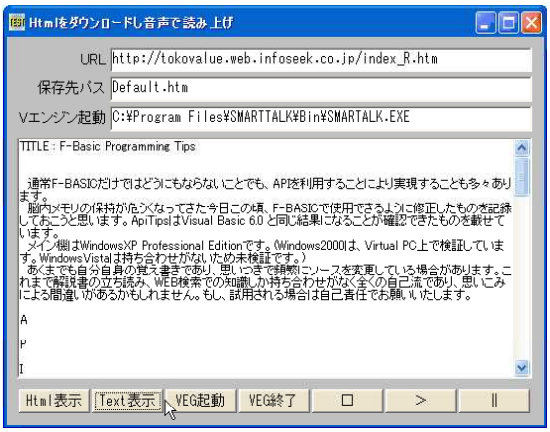



```
'=====================================
'= Urlダウンロード(II)
'=    (InternetOpen.bas)
'=====================================
#include "Windows.bi"

#define USER_AGENT "InternetReadFile Test" 'Header(任意)
#define INTERNET_OPEN_TYPE_DIRECT 1        'Proxyを経由せずにアクセスする
#define INTERNET_OPEN_TYPE_PROXY 3         'sProxyNameで指定したProxy経由でｱｸｾｽする
#define INTERNET_FLAG_RELOAD -2147483648   '
#define WI_READ_READSIZE 1024              'InternetReadFile で一度に読み込むサイズ
#define WI_INITBUFSIZE 32767               '

' インターネットのハンドルの作成
Declare Function Api_InternetOpen& Lib "wininet" Alias "InternetOpenA" (ByVal sAgent$,
ByVal lAccessType&, ByVal sProxyName$, ByVal sProxyBypass$, ByVal lFlags&)

' Win32インターネット関数のハンドルのクローズ
Declare Function Api_InternetCloseHandle% Lib "wininet" Alias "InternetCloseHandle"
(ByVal hInet&)

' インターネット上のファイルの読み込み
Declare Function Api_InternetReadFile% Lib "wininet" Alias "InternetReadFile" (ByVal
hFile&, ByVal sBuffer$, ByVal lNumBytesToRead&, lNumberOfBytesRead&)

' URLのオープン
Declare Function Api_InternetOpenUrl& Lib "wininet" Alias "InternetOpenUrlA" (ByVal
hInternetSession&, ByVal lpszUrl$, ByVal lpszHeaders$, ByVal dwHeadersLength&, ByVal
dwFlags&, ByVal dwContext&)
Var Shared Edit1 As Object
Var Shared Edit2 As Object
Var Shared Edit3 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 12
Edit3.Attach GetDlgItem("Edit3") : Edit3.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
```

```
Var Shared URL As String
Var Shared FileName As String
Var Shared Buff As String
Var Shared hFile As Byte

'=====================================
'=
'=====================================
Declare Sub Save_File edecl ()
Sub Save_File()
    hFile = FreeFile
    Open FileName For BinIO As hFile

    If Lof(hFile) <> 0 Then
        Close hFile
        Kill FileName
        Open FileName For BinIO As hFile
    End If

    FWrite hFile, Buff
    Close hFile
End Sub

'=====================================
'=
'=====================================
Declare Sub Html_Dsp edecl ()
Sub Html_Dsp()
    Var Word As String

    hFile = FreeFile
    Open FileName For BinInp As hFile
    Buff = Space$(Lof(hFile))
    FRead hFile, Buff
    Buff = JConv$(Buff, 0, 2)
    Edit2.SetWindowText Buff
    Close hFile
End Sub

'=====================================
'=
'=====================================

Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    URL = "http://tokovalue.jp/"
    FileName = "F:¥_FB_API_E¥__I¥test5.htm"

    Edit1.SetWindowText URL
    Edit3.SetWindowText FileName
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var nOpen As Long
    Var nFile As Long
    Var Leng As Integer
    Var ReadSize As Long
    Var Ret As Long

    URL = GetDlgItemText("Edit1")
    FileName = GetDlgItemText("Edit3")

    Leng = 32767
    Buff = Space$(Leng)

    nOpen = Api_InternetOpen("", INTERNET_OPEN_TYPE_DIRECT, ByVal 0, ByVal 0, 0)
```

```
    nFile = Api_InternetOpenUrl(nOpen, URL, ByVal 0, ByVal 0, INTERNET_FLAG_RELOAD, ByVal
0)

    Ret = Api_InternetReadFile(nFile, Buff, Leng, ReadSize)
    Buff = Left$(buff, ReadSize)

    Ret = Api_InternetCloseHandle(nFile)
    Ret = Api_InternetCloseHandle(nOpen)

    Save_File
    Html_Dsp
End Sub

'=======================================
'=
'=======================================
Declare Sub MainForm_Resize edecl ()
Sub MainForm_Resize()
    If GetWidth < 550 Or GetHeight < 370 Then
        SetWindowSize 550, 370
    End If

    Edit2.SetWindowSize GetWidth - 30, GetHeight - 144
    Button1.MoveWindow GetWidth - 88, GetHeight - 66
End Sub

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

URLダウンロード(III)読み上げ



Htmlをダウンロードし、フリーソフト(HtoX32c.exe)でTextに変換し、音声合成エンジン体験版(SMARTTALK)を使って読ませています。※音声合成エンジン体験版(SMARTTALK)は、下記からダウンロード　できます。

http://www.oki.com/jp/Cng/FTP/smart/st3trial.exe
注)ウィルスバスターを使っている場合、インストール時終了させる必要があるようです。

```
'=======================================
'= Urlダウンロード(III)読み上げ
'=    (InternetReadFile2.bas)
'=======================================
#include "Windows.bi"

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

' インターネットのハンドルの作成
Declare Function Api_InternetOpen& Lib "wininet" Alias "InternetOpenA" (ByVal sAgent$,
ByVal lAccessType&, ByVal sProxyName$, ByVal sProxyBypass$, ByVal lFlags&)

' Win32インターネット関数のハンドルのクローズ
Declare Function Api_InternetCloseHandle% Lib "wininet" Alias "InternetCloseHandle"
(ByVal hInet&)

' インターネット上のファイルの読み込み
Declare Function Api_InternetReadFile% Lib "wininet" Alias "InternetReadFile" (ByVal
hFile&, ByVal sBuffer$, ByVal lNumBytesToRead&, lNumberOfBytesRead&)

' URLのオープン
Declare Function Api_InternetOpenUrl& Lib "wininet" Alias "InternetOpenUrlA" (ByVal
hInternetSession&, ByVal lpszUrl$, ByVal lpszHeaders$, ByVal dwHeadersLength&, ByVal
dwFlags&, ByVal dwContext&)

' 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&)

' 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)

' カーソルの現在のスクリーン座標の取得
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&)

' ウィンドウのクラス名を取得する関数の宣言
Declare Function Api_GetClassName& Lib "user32" Alias "GetClassNameA" (ByVal hWnd&,
ByVal lpClassName$, ByVal nMaxCount&)

' 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
Declare Function Api_FindWindow& Lib "user32" Alias "FindWindowA" (ByVal lpClassName$,
ByVal lpWindowName$)

' 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&)

' 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&)

' 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)

' 指定されたクライアント座標を含む子ウインドウのハンドルを返す
Declare Function Api_ChildWindowFromPoint& Lib "user32" Alias "ChildWindowFromPoint"
(ByVal hWnd&, ByVal xPoint&, ByVal yPoint&)
#define USER_AGENT "InternetReadFile Test"'Header(任意)
```

```
#define INTERNET_OPEN_TYPE_DIRECT 1          '直接接続
#define INTERNET_OPEN_TYPE_PRECONFIG 0       'IEの設定に従い接続
#define INTERNET_OPEN_TYPE_PROXY 3           'Proxy経由接続
#define INTERNET_FLAG_RELOAD -2147483648     'ローカルのキャッシュを無視し、常にサーバからデータ取得
#define WI_READ_READSIZE 1024                'InternetReadFile で一度に読み込むサイズ
#define WI_INITBUFSIZE 32767                 '
#define WM_GETTEXT &HD                       'ｺﾝﾄﾛｰﾙのキャプション・テキストをバッファにコピー
#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_SHOW 5                            'ウィンドウをアクティブ化し現在の位置とｻｲｽﾞで表示
#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_CLICK &HF5                        'コマンドボタンをクリックした状態を擬似的に実現

Var Shared Edit(3) As Object
Var Shared Text(2) As Object
Var Shared Button(6) As Object

For i = 0 To 3
    Edit(i).Attach GetDlgItem("Edit" & Trim$(Str$(i + 1)))
    If i = 1 Then Edit(i).SetFontSize 12 Else Edit(i).SetFontSize 14
Next

For i = 0 To 6
    Button(i).Attach GetDlgItem("Button" & Trim$(Str$(i + 1)))
    Button(i).SetFontSize 14
Next

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

Var Shared URL As String
Var Shared FileName As String
Var Shared EnginePath As String
Var Shared Buff As String
Var Shared hFile As Byte
Var Shared hWnd As Long
Var Shared pt As POINTAPI

Declare Sub Html_Dsp edecl ()
Declare Sub Text_Dsp edecl ()
Declare Sub Save_File edecl ()

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    URL = "http://tokovalue.jp/index_R.htm"                      '既存のURL
    FileName = "Default.htm"                                     '既存のフォルダ
    EnginePath = "C:¥Program Files¥SMARTTALK¥Bin¥SMARTALK.EXE"   '既知のパス

    Edit(0).SetWindowText URL
    Edit(2).SetWindowText FileName
    Edit(3).SetWindowText EnginePath
End Sub

'=====================================
'=  HTML取得
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var nOpen As Long
    Var nFile As Long
    Var Leng As Integer
    Var ReadSize As Long
    Var Ret As Long

    URL = GetDlgItemText("Edit1")
    FileName = GetDlgItemText("Edit3")
```

```
    Leng = 32767
    Buff = Space$(Leng)

    nOpen = Api_InternetOpen("", INTERNET_OPEN_TYPE_DIRECT, ByVal 0, ByVal 0, 0)
    nFile = Api_InternetOpenUrl(nOpen, URL, ByVal 0, ByVal 0, INTERNET_FLAG_RELOAD, ByVal
0)

    Ret = Api_InternetReadFile(nFile, Buff, Leng, ReadSize)
    Buff = Left$(buff, ReadSize)

    Ret = Api_InternetCloseHandle(nFile)
    Ret = Api_InternetCloseHandle(nOpen)

    Save_File
    Html_Dsp
End Sub

'=====================================
'= HTML保存
'=====================================
Sub Save_File()
    hFile = FreeFile
    Open FileName For BinIO As hFile
    If Lof(hFile) <> 0 Then
        Close hFile
        Kill FileName
        Open FileName For BinIO As hFile
    End If
    FWrite hFile, Buff
    Close hFile
End Sub

'=====================================
'= HTML表示
'=====================================
Sub Html_Dsp()
    hFile = FreeFile
    Open FileName For BinInp As hFile
    Buff = Space$(Lof(hFile))
    FRead hFile, Buff
    Buff = JConv$(Buff, 0, 2)
    Edit(1).SetWindowText Buff
    Close hFile
End Sub

'=====================================
'= HTML → TEXT コンバーター（フリーソフト）
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Shell "HtoX32c.exe", FileName & " /O0 /F1", 0
    Text_DSP
End Sub

'=====================================
'= テキスト表示・文字列を全選択しクリップボードへ
'=====================================
Sub Text_Dsp()
    On Error Goto *ErrHandler

    Edit(1).SetWindowText ""
    FileName = Left$(FileName, Len(FileName) - 3) & "txt"

    hFile = FreeFile
    Open FileName For BinInp As hFile

    If Lof(hFile) <> 0 Then
        Buff = Space$(Lof(hFile))
        FRead hFile, Buff
        Buff = JConv$(Buff, 0, 2)
```

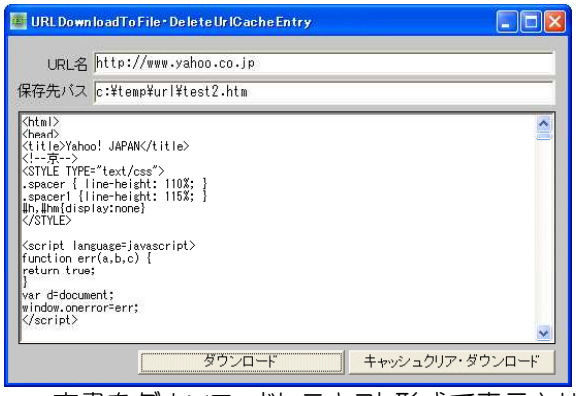
```
        Edit(1).SetWindowText Buff
        Close hFile
    End If

    Edit(1).SetSelText 0, -1
    Edit(1).Copy
    Wait 10
    Edit(1).SetSelText -1, -1
    Exit Sub

*ErrHandler
    Resume Next
End Sub

'=====================================
'= Voiceエンジン起動(既知)
'=====================================
Declare Sub Button3_on edecl ()
Sub Button3_on()
    Var Rct As RECT
    Var Ret As Long

    Shell EnginePath
    Wait 400

    hWnd = Api_FindWindow("ThunderRT5Form", "SMARTTALK")
    Ret = Api_ShowWindow(hWnd, SW_HIDE)

    If hWnd <> 0 Then
        Ret = Api_GetWindowRect(hWnd, Rct)
        Ret = Api_MoveWindow(hWnd, 0, 0, 348, 253, 1)
    End If
End Sub

'=====================================
'= Vエンジン終了
'=====================================

Declare Sub Button4_on edecl ()
Sub Button4_on()
    Var Ret As Long

    Ret = Api_SendMessage(hWnd, WM_CLOSE, 0, 0)
End Sub

'=====================================
'= StopButtonをクリック
'=====================================
Declare Sub Button5_on edecl ()
Sub Button5_on()
    Var hStopButton As Long
    Var Ret As Long

    pt.x = 20
    pt.y = 150

    hStopButton = Api_ChildWindowFromPoint(hWnd, pt.x, pt.y)
    Ret = Api_SendMessage(hStopButton, BM_CLICK, 0, 0)
End Sub

'=====================================
'= PlayButtonをクリック
'=====================================
Declare Sub Button6_on edecl ()
Sub Button6_on()
    Var hPlayButton As Long
    Var Ret As Long

    pt.x = 100
    pt.y = 150
```

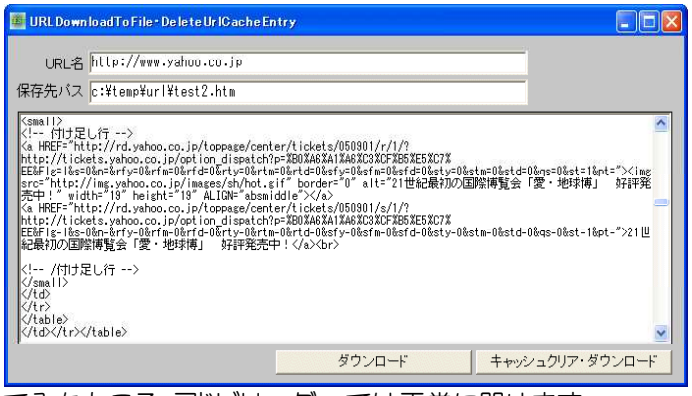
```
    hPlayButton = Api_ChildWindowFromPoint(hWnd, pt.x, pt.y)
    Ret = Api_SendMessage(hPlayButton, BM_CLICK, 0, 0)
End Sub

'=====================================
'= PauseButtonをクリック
'=====================================
Declare Sub Button7_on edecl ()
Sub Button7_on()
    Var hPauseButton As Long
    Var Ret As Long

    pt.x = 180
    pt.y = 150

    hPauseButton = Api_ChildWindowFromPoint(hWnd, pt.x, pt.y)
    Ret = Api_SendMessage(hPauseButton, BM_CLICK, 0, 0)
End Sub

'=====================================
'= フォームリサイズ
'=====================================
Declare Sub MainForm_Resize edecl ()
Sub MainForm_Resize()
    If GetWidth < 550 Or GetHeight < 410 Then
        SetWindowSize 550, 410
    End If
    Edit(1).SetWindowSize GetWidth - 30, GetHeight - 176
    Button(0).MoveWindow GetWidth - 538, GetHeight - 70
    Button(1).MoveWindow GetWidth - 464, GetHeight - 70
    Button(2).MoveWindow GetWidth - 390, GetHeight - 70
    Button(3).MoveWindow GetWidth - 316, GetHeight - 70
    Button(4).MoveWindow GetWidth - 242, GetHeight - 70
    Button(5).MoveWindow GetWidth - 168, GetHeight - 70
    Button(6).MoveWindow GetWidth - 94, GetHeight - 70
End Sub

'=====================================
'= フォームリサイズ
'=====================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
    Button4_on
    End
End Sub

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

URLダウンロードとキャッシュファイル削除

URLのHTML等をダウンロードしファイルに保存します。また、インターネット一時ファイル（キャッシュ）を削除します。
URLDownloadToFile URLのHTML等をダウンロードしファイルに保存
DeleteUrlCacheEntry インターネット一時ファイルの削除

Yahooのトップページをダウンロード、test2.htmとして保存し読込表示させています。
フォルダが存在しない場合はエラーになります。（エラー処理はしていません）





PDF文書をダウンロードしテキスト形式で表示させてみたところ。アドビリーダーでは正常に開けます。


```
'=======================================
'= URLダウンロードとキャッシュファイル削除
'=    (URLDownloadToFile.bas)
'=======================================
#include "Windows.bi"

' URLのHTML等をダウンロードしファイルに保存
Declare Function Api_URLDownloadToFile& Lib "urlmon" Alias "URLDownloadToFileA" (ByVal
pCaller&, ByVal szURL$, ByVal szFileName$, ByVal dwReserved&, ByVal lpfnCB&)

' インターネット一時ファイルの削除
Declare Function Api_DeleteUrlCacheEntry& Lib "Wininet" Alias "DeleteUrlCacheEntryA"
(ByVal lpszUrlName$)

#define ERROR_SUCCESS 0                    '正常終了の戻り値を示す
#define BINDF_GETNEWESTVERSION &H10        '

Var Shared Edit1 As Object
Var Shared Edit2 As Object
Var Shared Edit3 As Object
Var Shared Button1 As Object
Var Shared Button2 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 12
Edit2.Attach GetDlgItem("Edit2") : Edit2.SetFontSize 14
Edit3.Attach GetDlgItem("Edit3") : Edit3.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14

Var Shared sSourceUrl As String
Var Shared sLocalFile As String
Declare Sub HTML_DSP edecl ()

'=======================================
'=
'=======================================
Declare Function DownloadFile(sSourceUrl As String, sLocalFile As String) As Integer
Function DownloadFile(sSourceUrl As String, sLocalFile As String) As Integer

    DownloadFile = Api_URLDownloadToFile(0, sSourceUrl, sLocalFile,
BINDF_GETNEWESTVERSION, 0)
```

```
End Function

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    sSourceUrl = "http://tokovalue.jp/"
    sLocalFile = "test2.htm"

    Edit2.SetWindowText sSourceUrl
    Edit3.SetWindowText sLocalFile
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    sSourceUrl = Edit2.GetWindowText
    sLocalFile = Edit3.GetWindowText

    Edit1.SetWindowText ""
    Edit2.SetWindowText sSourceUrl
    Edit3.SetWindowText sLocalFile

    If DownloadFile(sSourceUrl, sLocalFile) = ERROR_SUCCESS Then HTML_DSP
End Sub

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

    sSourceUrl = Edit2.GetWindowText
    sLocalFile = Edit3.GetWindowText

    Edit1.SetWindowText ""
    Edit2.SetWindowText sSourceUrl
    Edit3.SetWindowText sLocalFile

    Ret = Api_DeleteUrlCacheEntry(sSourceUrl)
    txt = "保存済ファイルを読み込みます！"
    If Ret = 0 Then
        A% = MessageBox("", "エラー！" & Chr$(13) & Chr$(13) & txt, 0, 2)
    Else
        A% = MessageBox("", "キャッシュを削除しました！" & Chr$(13) & Chr$(13) & txt, 0, 2)
    End If

    If DownloadFile(sSourceUrl, sLocalFile) = ERROR_SUCCESS Then HTML_DSP
End Sub

'=====================================
'=
'=====================================
Sub HTML_DSP()
    Var hfile As byte
    Var Buff As String

    hfile = FreeFile
    Open sLocalFile For BinInp As hfile
    Buff = Space$(lof(hfile))
    FRead hfile, Buff
    Buff = JConv$(Buff, 0, 2)
    Edit1.SetWindowText Buff
    Close hfile
End Sub
```

```
'=====================================
'=
'=====================================
Declare Sub MainForm_Resize edecl ()
Sub MainForm_Resize()
    If GetWidth < 550 Or GetHeight < 370 Then
        SetWindowSize 550, 370
    End If

    Edit1.SetWindowSize GetWidth - 30, GetHeight - 144
    Button1.MoveWindow GetWidth - 424, GetHeight - 66
    Button2.MoveWindow GetWidth - 220, GetHeight - 66
End Sub

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

URLに接続できるかチェック(Ⅰ)

InternetCheckConnection インターネットに接続できるかどうかをチェックします。

例では、最初に正規のURLを、次に間違ったURLを設定してチェックしています。
※Internet Explorer 3.0以降


```
'=====================================
'= URLに接続できるかチェック
'=    (InternetCheckConnection.bas)
'=====================================
#define FLAG_ICC_FORCE_CONNECTION &H1

' インターネットへの接続することができるか調べる
Declare Function Api_InternetCheckConnection& Lib "wininet" Alias
"InternetCheckConnectionA" (ByVal lpszUrl$, ByVal dwFlags&, ByVal dwReserved&)

Var Url As String

Url = "http://tokovalue.jp/"
GoSub *Check
Print
Url = "http://tokovalue.jo/"
GoSub *Check

Stop
End

*Check
If Api_InternetCheckConnection(Url, FLAG_ICC_FORCE_CONNECTION, 0) = 0 Then
    Print Url & " に接続できません！"
Else
    Print Url & " に接続できます！"
End If
return
```

URLに接続できるかチェック(II)

InternetCheckConnection インターネットへの接続することができるか調べる


```
'=====================================
'= URLに接続できるかチェック
'=    (InternetCheckConnection.bas)
'=====================================
#include "Windows.bi"

#define FLAG_ICC_FORCE_CONNECTION &H1

' インターネットへの接続することができるか調べる
Declare Function Api_InternetCheckConnection& Lib "wininet" Alias
"InternetCheckConnectionA" (ByVal lpszUrl$, ByVal dwFlags&, ByVal dwReserved&)

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 Function CheckInetConnection(sUrlOfInternet As String) As Integer
Function CheckInetConnection(sUrlOfInternet As String) As Integer

    CheckInetConnection = Api_InternetCheckConnection(sUrlOfInternet,
FLAG_ICC_FORCE_CONNECTION, 0)

End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var sUrl As String

    sUrl = Edit1.GetWindowtext

    If CheckInetConnection(sUrl) Then
        Text1.SetWindowtext sUrl & " と、接続可能です。"
    Else
        Text1.SetWindowtext sUrl & " と、接続できません。"
    End If
End Sub

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

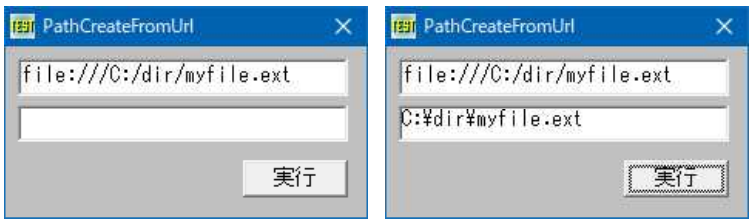
URLの結合

InternetCombineUrl 絶対パスと相対パスを組み合わせて新しいURLを作成します。


```
'=====================================
'= URLの結合
'=    (InternetCombineUrl.bas)
'=====================================

' 絶対パスと相対パスを組み合わせて新しいURLを作成
Declare Sub Api_InternetCombineUrl Lib "wininet" Alias "InternetCombineUrlA" (ByVal
lpszBaseUrl$, ByVal lpszRelativeUrl$, ByVal lpszBuffer$, ByRef lpdwBufferLength&, ByVal
dwFlags&)

#define ICU_BROWSER_MODE &H2000000          '#や?の後のキャラクタを変換しない
#define ICU_DECODE &H10000000               '%XX形式のエスケープシーケンスをキャラクタに変換
#define ICU_ENCODE_PERCENT &H1000           '%文字を変換する
#define ICU_ENCODE_SPACES_ONLY &H4000000    'スペースのみエンコード
#define ICU_ESCAPE -2147483648              '特殊文字をエスケープシーケンスに変換する
#define ICU_NO_ENCODE &H20000000            '特殊文字をエスケープシーケンスに変換しない
#define ICU_NO_META &H8000000               '.や..等、メタシーケンスを削除しない
#define ICU_USERNAME &H40000000             'ユーザ名追加時に、ログイン時に指定されたユーザ名を使
                                            用する

Var Buff As String
Var str1 As String
Var str2 As String

str1 = "http://tokovalue.jp"
str2 = "/index.htm"

Buff = String$(255, Chr$(0))

Api_InternetCombineUrl str1, str2, Buff, 255, ICU_ENCODE_PERCENT

strBuffer = Left$(Buff, InStr(Buff, Chr$(0)) - 1)

Print Buff

Stop
End
```

URLパスを取得しDOSパスに変換

PathCreateFromUrl URLパスを取得しDOSパスに変換

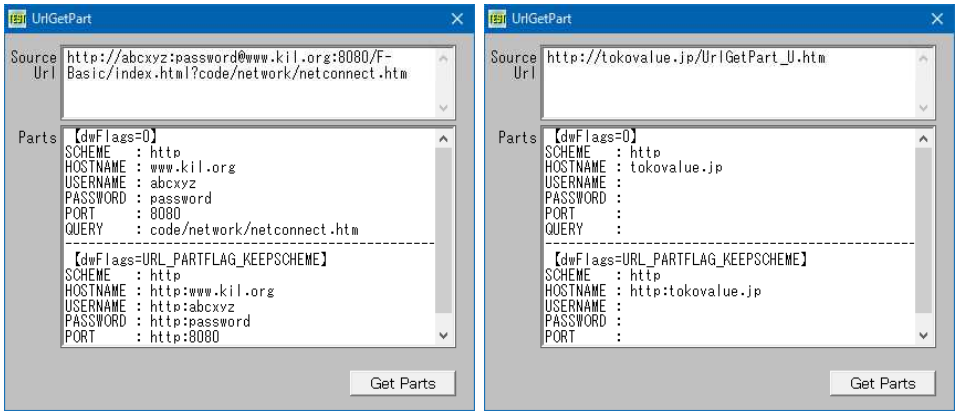



```
'=====================================
'= URLパスを取得しDOSパスに変換
'=    (PathCreateFromUrl.bas)
'=====================================
#include "Windows.bi"
```

```
' 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&)

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 "file:///C:/dir/myfile.ext"
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Buff As String
    Var sSave As String

    Buff = String$(100, 0)
    sSave = Edit1.GetWindowText

    Api_PathCreateFromUrl sSave, Buff, Len(Buff), 0

    Text1.SetWindowText TrimNull(Buff)
End Sub

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

URL文字列のパーツを取得

URL文字列をパーツに分解します。
UrlGetPart 指定のURLのパーツを取得




```
'=======================================
'= URL文字列のパーツを取得
'=    (UrlGetPart.bas)
'=======================================
#include "Windows.bi"

#define MAX_PATH 260
#define ERROR_SUCCESS 0                    '正常終了の戻り値を示す
#define URL_PART_SCHEME 1                  'スキーム
#define URL_PART_HOSTNAME 2                'ホストネーム
#define URL_PART_USERNAME 3                'ユーザーネーム
#define URL_PART_PASSWORD 4                'パスワード
#define URL_PART_PORT 5                    'ポート
#define URL_PART_QUERY 6                   'クエリ
#define URL_PARTFLAG_KEEPSCHEME &H1

' 指定のURLのパーツを取得
Declare Function Api_UrlGetPart& Lib "shlwapi" Alias "UrlGetPartA" (ByVal pszIn$, ByVal
pszOut$, pcchOut&, ByVal dwPart&, ByVal dwFlags&)

Var Shared strURL As String
Var Shared Text1 As Object
Var Shared Text2 As Object
Var Shared Edit1 As Object
Var Shared List1 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Text2.Attach GetDlgItem("Text2") : Text2.SetFontSize 14
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=======================================
'=
'=======================================
Declare Function GetUrlParts(sUrl As String, dwPart As Long, dwFlags As Long) As String
Function GetUrlParts(sUrl As String, dwPart As Long, dwFlags As Long) As String
    Var sPart As String
    Var dwSize As Long

    If Len(sUrl) > 0 Then
        sPart = Space$(MAX_PATH)
        dwSize = Len(sPart)

        If Api_UrlGetPart(sUrl, sPart, dwSize, dwPart, dwFlags) = ERROR_SUCCESS Then
            GetUrlParts = Left$(sPart, dwSize)
        End If
    End If
End Function

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

```
Sub Button1_on()
    strURL = Edit1.GetWindowText

    List1.ResetContent
    List1.AddString "【dwFlags=0】"
    List1.AddString "SCHEME   : " & GetUrlParts(strURL, URL_PART_SCHEME, 0)
    List1.AddString "HOSTNAME : " & GetUrlParts(strURL, URL_PART_HOSTNAME, 0)
    List1.AddString "USERNAME : " & GetUrlParts(strURL, URL_PART_USERNAME, 0)
    List1.AddString "PASSWORD : " & GetUrlParts(strURL, URL_PART_PASSWORD, 0)
    List1.AddString "PORT     : " & GetUrlParts(strURL, URL_PART_PORT, 0)
    List1.AddString "QUERY    : " & GetUrlParts(strURL, URL_PART_QUERY, 0)

    List1.AddString String$(50, "-")
    List1.AddString "【dwFlags=URL_PARTFLAG_KEEPSCHEME】"
    List1.AddString "SCHEME   : " & GetUrlParts(strURL, URL_PART_SCHEME,
URL_PARTFLAG_KEEPSCHEME)
    List1.AddString "HOSTNAME : " & GetUrlParts(strURL, URL_PART_HOSTNAME,
URL_PARTFLAG_KEEPSCHEME)
    List1.AddString "USERNAME : " & GetUrlParts(strURL, URL_PART_USERNAME,
URL_PARTFLAG_KEEPSCHEME)
    List1.AddString "PASSWORD : " & GetUrlParts(strURL, URL_PART_PASSWORD,
URL_PARTFLAG_KEEPSCHEME)
    List1.AddString "PORT     : " & GetUrlParts(strURL, URL_PART_PORT,
URL_PARTFLAG_KEEPSCHEME)
    List1.AddString "QUERY    : " & GetUrlParts(strURL, URL_PART_QUERY,
URL_PARTFLAG_KEEPSCHEME)
End Sub

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

WaveAudioの出力情報を取得

ウェーブオーディオの出力に関する情報を取得します。
waveOutGetNumDevs システムに存在するWave出力デバイスの数を取得
waveOutGetDevCaps ウエーブオーディオ出力デバイスの性能を取得

WindowsXP機

Windows200機

Windows10機


```
'=====================================
'= WaveAudioの出力情報を取得
'=    (waveOutGetDevCaps.bas)
'=====================================
#include "Windows.bi"

Type WAVEOUTCAPS
    wMid                As Integer              '機器のメーカーのメーカー識別子
```

```
    wPid            As Integer          'Ｍ機器の製品識別子
    vDriverVersion  As Long             'Ｍ機器のバージョン番号
    szPname         As String * 32      'Ｍ製品名
    dwFormats       As Long             'Ｍサポートされる標準のフォーマット(WAVE_FORMAT_...)
    wChannels       As Integer          'Ｍオーディオのチャンネルの数(1:Mono 2:Stereo)
    dwSuppor        As Long             'Ｍオプションの機能
End Type

'Ｍ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" ()

'Ｍウェーブフォームオーディオ出力デバイスの性能を取得
Declare Function Api_waveOutGetDevCaps& Lib "winmm" Alias "waveOutGetDevCapsA" (ByVal
uDeviceID&, lpCaps As WAVEOUTCAPS, ByVal uSize&)

#define WAVE_MAPPER -1

#define WAVE_FORMAT_1M08 &H1            '11.025 kHz Mono 8-bit
#define WAVE_FORMAT_1M16 &H4            '11.025 kHz Mono 16-bit
#define WAVE_FORMAT_1S08 &H2            '11.025 kHz Stereo 8-bit
#define WAVE_FORMAT_1S16 &H8            '11.025 kHz Stereo 16-bit
#define WAVE_FORMAT_2M08 &H10           '22.050 kHz Mono 8-bit
#define WAVE_FORMAT_2M16 &H40           '22.050 kHz Mono 16-bit
#define WAVE_FORMAT_2S08 &H20           '22.050 kHz Stereo 8-bit
#define WAVE_FORMAT_2S16 &H80           '22.050 kHz Stereo 16-bit
#define WAVE_FORMAT_4M08 &H100          '44.100 kHz Mono 8-bit
#define WAVE_FORMAT_4M16 &H400          '44.100 kHz Mono 16-bit
#define WAVE_FORMAT_4S08 &H200          '44.100 kHz Stereo 8-bit
#define WAVE_FORMAT_4S16 &H800          '44.100 kHz Stereo 16-bit

#define WAVECAPS_LRVOLUME &H8           '左右2チャンネル
#define WAVECAPS_PITCH &H1              '
#define WAVECAPS_PLAYBACKRATE &H2       '
#define WAVECAPS_SYNC &H10              '
#define WAVECAPS_VOLUME &H4             '
#define WAVECAPS_SAMPLEACCURATE &H20    '

'メーカー識別子
#define MM_ANTEX 31                     'Antex Electronics Corporation
#define MM_APPS 42                      'APPS Software
#define MM_APT 56                       'Audio Processing Technology
#define MM_ARTISOFT 20                  'Artisoft, Inc.
#define MM_AST 64                       'AST Research, Inc.
#define MM_ATI 27                       'ATI Technologies, Inc.
#define MM_AUDIOFILE 47                 'Audio, Inc.
#define MM_AUDIOPT 74                   'Audio Processing Technology
#define MM_AURAVISION 80                'Auravision Corporation
#define MM_AZTECH 52                    'Aztech Labs, Inc.
#define MM_CANOPUS 49                   'Canopus, Co., Ltd.
#define MM_CAT 41                       'Computer Aided Technology, Inc.
#define MM_COMPUSIC 89                  'Compusic
#define MM_COMPUTER_FRIENDS 45          'Computer Friends, Inc.
#define MM_CONTROLRES 84                'Control Resources Corporation
#define MM_CREATIVE 2                   'Creative Labs, Inc.
#define MM_DIALOGIC 93                  'Dialogic Corporation
#define MM_DOLBY 78                     'Dolby Laboratories, Inc.
#define MM_DSP_GROUP 43                 'DSP Group, Inc.
#define MM_DSP_SOLUTIONS 25             'DSP Solutions, Inc.
#define MM_ECHO 39                      'Echo Speech Corporation
#define MM_EPSON 50                     'Seiko Epson Corporation, Inc.
#define MM_ESS 46                       'ESS Technology, Inc.

#define MM_APT_ACE100CD 1               'ACE 100 CD
#define MM_ARTISOFT_SBWAVEIN 1          'Artisoft Sounding Board In
#define MM_ARTISOFT_SBWAVEOUT 2         'Artisoft Sounding Board Out
#define MM_AZTECH_AUX_CD 401            'Aztech Auxiales CD-Audio
#define MM_AZTECH_AUX_LINE 402          'Aztech Auxialer Line In
#define MM_AZTECH_AUX_MIC 403           'Aztech Auxialer Microphone In
#define MM_AZTECH_AUX 404               'Aztech Auxialer In
#define MM_AZTECH_DSP16_WAVEIN 65       'Aztech DSP 16 Wave In
```

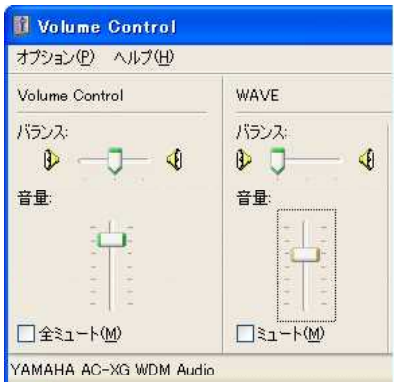
```
#define MM_AZTECH_DSP16_WAVEOUT 66          'Aztech DSP 16 Wave Out
#define MM_AZTECH_DSP16_FMSYNTH 68          'Aztech DSP 16 FM Synthesizer
#define MM_AZTECH_DSP16_WAVESYNTH 70        'Aztech DSP 16 Wave Synthesizer
#define MM_AZTECH_NOVA16_WAVEIN 71          'Aztech Nova 16 Wave In
#define MM_AZTECH_NOVA16_WAVEOUT 72         'Aztech Nova 16 Wave Out
#define MM_AZTECH_NOVA16_MIXER 73           'Aztech Nova 16 Mixer
#define MM_AZTECH_PRO16_WAVEIN 33           'Aztech Pro 16 Wave In
#define MM_AZTECH_PRO16_WAVEOUT 34          'Aztech Pro 16 Wave Out
#define MM_AZTECH_WASH16_WAVEIN 74          'Aztech Wash 16 Wave In
#define MM_AZTECH_WASH16_WAVOUT 75          'Aztech Wash 16 Wave Out
#define MM_AZTECH_WASH16_MIXER 76           'Aztech Wash 16 Mixer
#define MM_AZTECH_MIDIOUT 3                 'Aztech Midi Out
#define MM_AZTECH_MIDIIN 4                  'Aztech Midi In
#define MM_AZTECH_WAVEIN 17                 'Aztech Wave In
#define MM_AZTECH_WAVEOUT 18                'Aztech Wave Out
#define MM_AZTECH_FMSYNTH 20                'Aztech FM Synthesizer
#define MM_AZTECH_MIXER 21                  'Aztech Mixer
#define MM_CAT_WAVEOUT 1                    'Comuter Aided Technologies Wave Out
#define MM_CREATIVE_AUX_CD 401              'Creative Labs Soundblaster Pro CD In
#define MM_CREATIVE_AUX_LINE 402            'Creative Labs Soundblaster Pro Line In
#define MM_CREATIVE_AUX_MIC 403             'Creative Labs Soundblaster Pro Microphone
                                In
#define MM_CREATIVE_AUX_MASTER 404          'Creative Labs Soundblaster Pro Master
#define MM_CREATIVE_AUX_PCSPK 405           'Creative Labs Soundblaster Pro PC-Speaker
#define MM_CREATIVE_AUX_WAVE 406            'Creative Labs Soundblaster Pro Wave
#define MM_CREATIVE_AUX_MIDI 407            'Creative Labs Soundblaster pro Midi
                                           Synthesizer
#define MM_CREATIVE_SB15_WAVEIN 1           'Creative Labs Soundblaster 1.5 Wave In
#define MM_CREATIVE_SB15_WAVEOUT 101        'Creative Labs Soundblaster 1.5 Wave Out
#define MM_CREATIVE_SB16_MIXER 409          'Creative Labs Soundblaster Pro 16 Mixer
#define MM_CREATIVE_SB20_WAVEIN 2           'Creative Labs Soundblaster 2.0 Wave In
#define MM_CREATIVE_SB20_WAVEOUT 102        'Creative Labs Soundblaster 2.0 Wave Out
#define MM_CREATIVE_SBP16_WAVEIN 4          'Creative Labs Soundblaster Pro 16 Wave In
#define MM_CREATIVE_SBP16_WAVEOUT 104       'Creative Labs Soundblaster Pro 16 Wave Out
#define MM_CREATIVE_SBPRO_WAVEIN 3          'Creative Labs Soundblaster Pro Wave In
#define MM_CREATIVE_SBPRO_WAVEOUT 103       'Creative Labs Soundblaster Pro Wave Out
#define MM_CREATIVE_SBPRO_MIXER 408         'Creative Labs Soundblaster Pro Mixer
#define MM_CREATIVE_MIDIOUT 201             'Creative Labs Soundblaster MIDI Out
#define MM_CREATIVE_MIDIIN 202              'Creative Labs Soundblaster MIDI In
#define MM_CREATIVE_FMSYNTH_MONO 301        'Creative Labs Soundblaster MONO FM
                                           Synthesizer
#define MM_CREATIVE_FMSYNTH_STEREO 302      'Creative Labs Soundblaster STEREO FM
                                           Synthesizer
#define MM_CREATIVE_MIDI_AWE32 303          'Creative Labs Soundblaster AWE 32 MIDI
#define MM_DSP_GROUP_TRUESPEECH &H1         'DSP Group True Speech
#define MM_DSP_SOLUTIONS_WAVEOUT 1          'DSP Solutions Wave Out
#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 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 woc As WAVEOUTCAPS
    Var MajorVer As Long
    Var MinorVer As Long
    Var Ret As Long

    List1.Resetcontent

    For i = WAVE_MAPPER To Api_waveOutGetNumDevs
        Ret = Api_waveOutGetDevCaps(i, woc, Len(woc))

        If Ret <> 0 And i = WAVE_MAPPER Then
```

```
            List1.AddString "Wave Mapper はありません！"
            List1.AddString "-------------------------------------" & vbCrLf & vbCrLf
        Else
            MajorVer = (woc.vDriverVersion And &HFF00) / &H100
            MinorVer = woc.vDriverVersion And &HFF

            List1.AddString "製品名: " & woc.szPname
            List1.AddString "Driver Version: " & Str$(MajorVer) & "." & Str$(MinorVer)
            List1.AddString "Audio Channel: " & Str$(woc.wChannels)

            If woc.wPid = MM_CREATIVE_SBP16_WAVEOUT Then
                List1.AddString "Type: Soundblaster Pro 16 WaveOut Compatible"
            End If

            If woc.wMid = MM_CREATIVE Then
                List1.AddString "メーカー: Creative Labs, Inc."
            End If

            If CInt(woc.dwFormats And WAVE_FORMAT_4S16) <> 0 Then
                List1.AddString "Format (01): 44.100 kHz - 16 Bit in Stereo"
            End If

            If (woc.dwSupport And WAVECAPS_LRVOLUME) = 0 Then
                List1.AddString "2チャンネルステレオをサポート"
            End If

            List1.AddString "-------------------------------------" & vbCrLf
        End If
    Next i
End Sub

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

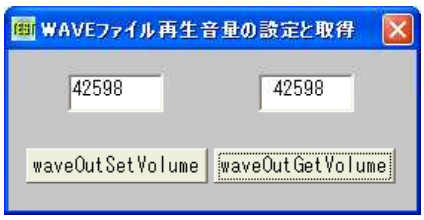
WAVEファイル再生音量の設定と取得

---

waveOutSetVolume WAVEファイル再生音量の設定
waveOutGetVolume WAVEファイル再生音量の取得



左:Volume Controlを図のように設定し••
右:その値を取得 。または、左側EditBoxに数値を入れ、••GetVolumeで
その値を取得します。

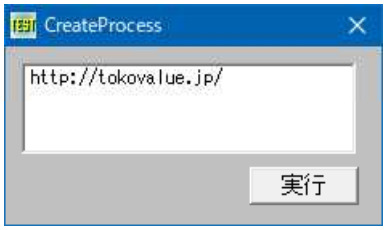



```
'====================================
'= WAVEファイル再生音量の設定と取得
'=    (waveOutGetVolume.bas)
'====================================
#include "Windows.bi"

' WAVEファイル再生音量の設定
Declare Function Api_waveOutSetVolume& Lib "Winmm" Alias "waveOutSetVolume" (ByVal
```

```
wDeviceID&, ByVal dwVolume&)

' WAVEファイル再生音量の取得
Declare Function Api_waveOutGetVolume& Lib "Winmm" Alias "waveOutGetVolume" (ByVal
wDeviceID&, dwVolume&)

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

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

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

    vol = Edit(0).GetWindowText
    tmp = Right$(Hex$((Val(vol) + 65536)), 4)
    vol = Str$(Val("&H" & tmp))
    Ret = Api_waveOutSetVolume(0, Val(vol))
End Sub

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

    Ret = Api_waveOutGetVolume(0, vol)
    tmp = "&H" & Right$(Hex$(vol), 4)
    Edit(1).SetWindowText Str$(Val(tmp))
End Sub

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

Webの起動

CreateProcess プロセスの起動
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
FindExecutable ファイル名に関連付けられている実行可能ファイル名とハンドルを取得
GetTempPath 一時フォルダ（テンポラリフォルダ）を取得



```
'=====================================
'= Webの起動
'=    (CreateProcess.bas)
'=====================================
#include "Windows.bi"

#define CREATE_NEW_CONSOLE &H10
#define NORMAL_PRIORITY_CLASS &H20          '通常クラス(一般的なプロセス)
#define INFINITE &HFFFF                     '無限に中断
#define STARTF_USESHOWWINDOW &H1            'wShowWindowを使用する
#define SW_SHOWNORMAL 1                     'SW_RESTOREと同じ
#define MAX_PATH 260
#define ERROR_FILE_NO_ASSOCIATION 31
#define ERROR_FILE_NOT_FOUND &H2            '指定されたファイルが見つからない
#define ERROR_PATH_NOT_FOUND &H3            '指定されたパスが見つからない
#define ERROR_FILE_SUCCESS 32
#define ERROR_BAD_FORMAT &HB                '間違ったフォーマットのプログラムを読み込もうとした

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 Name$,
ByVal Cmd$, pAtr&, tAtr&, ByVal iHand&, ByVal Flg&, ByVal Env&, ByVal Dir&, sInf As
STARTUPINFO, pInf As PROCESS_INFORMATION)

' 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&)

' ファイル名に関連付けられている実行可能ファイル名とハンドルを取得
Declare Function Api_FindExecutable& Lib "shell32" Alias "FindExecutableA" (ByVal
lpFile$, ByVal lpDirectory$, ByVal lpResult$)

' 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$)
```

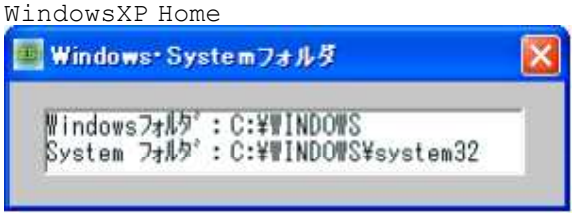
```
Var Shared Edit1 As Object
Var Shared Button1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 12
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
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 Function GetTempDir() As String
Function GetTempDir() As String
    Var tmp As String
    Var Ret As Long

    tmp = Space$(MAX_PATH)
    Ret = Api_GetTempPath(Len(tmp), tmp)

    GetTempDir = TrimNull(tmp)
End Function

'=====================================
'=
'=====================================
Declare Function GetBrowserName(dwFlagReturned As Long) As String
Function GetBrowserName(dwFlagReturned As Long) As String
    Var hFile As Long
    Var sResult As String
    Var sTempFolder As String

    'tmpフォルダを取得
    sTempFolder = GetTempDir()

    'ダミーのhtmlファイルを作成
    hFile = FreeFile
        Open sTempFolder & "dummy.html" For Output As #hFile
    Close #hFile

    '関連ファイルパスを取得
    sResult = Space$(MAX_PATH)
    dwFlagReturned = Api_FindExecutable("dummy.html", sTempFolder, sResult)

    '削除
    Kill sTempFolder & "dummy.html"

    GetBrowserName = TrimNull(sResult)
End Function

'=====================================
'=
'=====================================
Declare Function BuildCommandLine(sBrowser As String) As String
Function BuildCommandLine(sBrowser As String) As String
    sBrowser = LCase$(sBrowser)
```

```
    'internet explorer
    If InStr(sBrowser, "iexplore.exe") > 0 Then
        BuildCommandLine = " -nohome "

    'netscape 4.x
    Else If InStr(sBrowser, "netscape.exe") > 0 Then
        BuildCommandLine = " "

    'netscape 7.x
    Else If InStr(sBrowser, "netscp.exe") > 0 Then
        BuildCommandLine = " -url "
    Else
        BuildCommandLine = " "
    End If
End Function

'=====================================
'=
'=====================================
Declare Function StartNewBrowser(sURL As String) As Integer
Function StartNewBrowser(sURL As String) As Integer
    Var success As Long
    Var hProcess As Long
    Var sBrowser As String
    Var si As STARTUPINFO
    Var pi As PROCESS_INFORMATION
    Var sCmdLine As String

    sBrowser = GetBrowserName(success)

    If success >= ERROR_FILE_SUCCESS Then
        sCmdLine = BuildCommandLine(sBrowser)

        si.cb = Len(si)
        si.dwFlags = STARTF_USESHOWWINDOW
        si.wShowWindow = SW_SHOWNORMAL

        success = Api_CreateProcess(sBrowser, sCmdLine & sURL, ByVal 0, ByVal 0, ByVal 0,
NORMAL_PRIORITY_CLASS, ByVal 0, ByVal 0, si, pi)

        StartNewBrowser = pi.hProcess <> 0
        Ret = Api_CloseHandle(pi.hProcess)
        Ret = Api_CloseHandle(pi.hThread)
    End If
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var sURL As String

    sURL = Edit1.GetWindowText

    If Not StartNewBrowser(sURL) Then
        A% = MessageBox("", "見つかりません！", 0, 2)
    End If
End Sub

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

Windows・Systemフォルダの取得

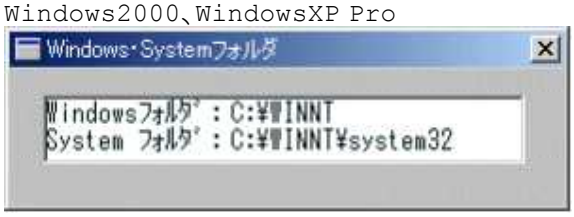
OSによりWindowsおよびSystemフォルダが異なります。
GetWindowsDirectory Windowsディレクトリのパス名を取得
GetSystemDirectory Windows のシステムディレクトリのパスを取得

WindowsXP Home

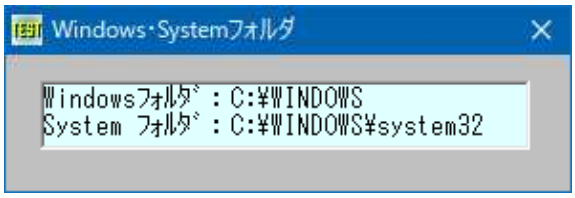


Windows98

Windows2000、WindowsXP Pro



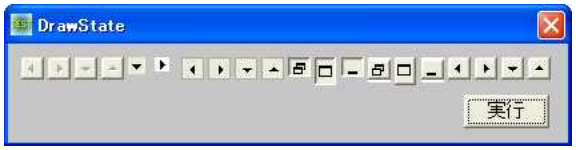
Windows10

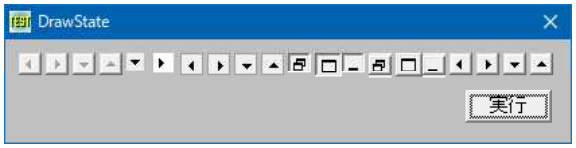



```
'=====================================
'= ウィンドウズ・システムフォルダ取得
'=    (WIN_SYS_FOLDER.bas)
'=====================================
#include "Windows.bi"

' 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&)

' 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&)

Var Shared Text1 As Object
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var WinPath As String * 255
    Var SysPath As String * 255
    Var WinPathLen As Long
    Var SysPathLen As Long
    Var txt As String

    WinPathLen = Api_GetWindowsDirectory(WinPath, Len(WinPath))
    SysPathLen = Api_GetSystemDirectory(SysPath, Len(SysPath))

    txt = "Windowsﾌｫﾙﾀﾞ:" & Left$(WinPath, InStr(WinPath, Chr$(0)) - 1) & Chr$(13,10)
    txt = txt & "System ﾌｫﾙﾀﾞ:" & Left$(SysPath, InStr(SysPath, Chr$(0)) - 1)

    Text1.SetWindowText txt
End Sub

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

Windows組込ビットマップを表示

Windowsに組み込まれているビットマップ(OBM_LFARROWI:32734～OBM_UPARROW:32753)を表示します。
DrawState イメージを加工して表示する
LoadImage 画像ファイルの読み込み






```
'=====================================
'= Windows組込ビットマップを表示
'=    (DrawState.bas)
'=====================================
#include "Windows.bi"

' イメージを加工して表示する
Declare Function Api_DrawState& Lib "user32" Alias "DrawStateA" (ByVal hDC&, ByVal hbr&,
ByVal lpOutputFunc&, ByVal lData&, ByVal wData&, ByVal X&, ByVal Y&, ByVal cx&, ByVal cy&,
ByVal fuFlags&)

' 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&)

' 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 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 DST_BITMAP &H4                  'ビットマップ
#define LR_DEFAULTSIZE &H40             '標準サイズで表示

' Windows組込ビットマップ(32734～32753)
#define OBM_LFARROWI 32734
#define OBM_RGARROWI 32735
#define OBM_DNARROWI 32736
#define OBM_UPARROWI 32737
#define OBM_COMBO    32738
#define OBM_MNARROW  32739
#define OBM_LFARROWD 32740
#define OBM_RGARROWD 32741
#define OBM_DNARROWD 32742
#define OBM_UPARROWD 32743
#define OBM_RESTORED 32744
#define OBM_ZOOMD    32745
#define OBM_REDUCED  32746
#define OBM_RESTORE  32747
#define OBM_ZOOM     32748
#define OBM_REDUCE   32749
#define OBM_LFARROW  32750
#define OBM_RGARROW  32751
#define OBM_DNARROW  32752
#define OBM_UPARROW  32753

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

```
    Var hImage As Long
    Var x As Integer
    Var OBM As Integer

    hDC = Api_GetDC(GethWnd)
    x = 10
    For OBM = 32734 To 32753
        hImage = Api_LoadImage(0, OBM, IMAGE_BITMAP, 0, 0, LR_DEFAULTSIZE)
        Ret = Api_DrawState(hDC, 0, 0, hImage, 0, x, 10, 0, 0, DST_BITMAP)
        x = x + 20
    Next
        Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

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

Windowsの終了・再起動(Win95/98)

Windowsの『ログオフ』『シャットダウン』『再起動』を実行します。
ExitWindowsEx Windowsの強制終了

Windows95およびWindows98でのみ実行できます。

```
'=====================================
'= Windows(Windows95/Windows98)の終了
'=   (ExitWindowsEx2.bas)
'=====================================
#include "Windows.bi"

' Windowsを強制終了
Declare Function Api_ExitWindowsEx& Lib "user32" Alias "ExitWindowsEx" (ByVal uFlags&,
ByVal dwReserved&)

#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

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 i

'=====================================
'=  ログオフ
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    Ret = Api_ExitWindowsEx(EWX_LOGOFF, 0)
End Sub
```

```
'=====================================
'=  シャットダウン
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Ret As Long

    Ret = Api_ExitWindowsEx(EWX_SHUTDOWN, 0)
End Sub

'=====================================
'=  リブート
'=====================================
Declare Sub Button3_on edecl ()
Sub Button3_on()
    Var Ret As Long

    Ret = Api_ExitWindowsEx(EWX_REBOOT, 0)
End Sub

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

Windowsの強制終了・再起動・ログオフ

呼び出し側プロセスはAdjustTokenPrivileges関数を呼び出し特権を有効にし、Windowsの『ログオフ』『シャットダウン』『再起動』を実行します。(WindowsNT以降)
GetCurrentProcess 現在のプロセスに対応する疑似ハンドルを取得
OpenProcessToken プロセスと結びつけられたアクセストークンを開く
ExitWindowsEx Windowsを強制終了
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
LookupPrivilegeValue 指定されたシステムで使われているローカル一意識別子(LUID)を取得し、指定された特権名をローカルで表現
AdjustTokenPrivileges 指定したアクセストークン無いの特権を有効または無効に設定


```
'=====================================
'= Windowsの終了・再起動・ログオフ
'= WindowsNT以降(Windows9xは対象外)
'=    (LookupPrivilegeValue.bas)
'=====================================
#include "Windows.bi"

#define ANYSIZE_ARRAY 1
#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
#define SE_PRIVILEGE_ENABLED &H2            '権限を有効にする
#define TOKEN_ADJUST_PRIVILEGES &H20        '特権を変更
#define TOKEN_QUERY &H8                     '問い合わせ
#define VER_PLATFORM_WIN32_NT 2             'WINDOWSNT、2000、XP
```

```
Type OSVERSIONINFO
    dwOSVersionInfoSize     As Long
    dwMajorVersion          As Long
    dwMinorVersion          As Long
    dwBuildNumber           As Long
    dwPlatformId            As Long
    szCSDVersion            As String * 128
End Type

Type LUID
    LowPart    As Long
    HighPart   As Long
End Type

Type LUID_AND_ATTRIBUTES
    pLuid             As LUID
    Attributes        As Long
End Type

Type TOKEN_PRIVILEGES
    PrivilegeCount                  As Long
    Privileges(ANYSIZE_ARRAY)       As LUID_AND_ATTRIBUTES
End Type

' 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" ()

' プロセスと結び付けられたアクセストークンを開く
Declare Function Api_OpenProcessToken& Lib "advapi32" Alias "OpenProcessToken" (ByVal
ProcessHandle&, ByVal DesiredAccess&, TokenHandle&)

' Windowsを強制終了
Declare Function Api_ExitWindowsEx& Lib "user32" Alias "ExitWindowsEx" (ByVal uFlags&,
ByVal dwReserved&)

' 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)

' 指定されたシステムで使われているローカル一意識別子(LUID)を取得し、指定された特権名をローカルで表現
Declare Function Api_LookupPrivilegeValue& Lib "advapi32" Alias "LookupPrivilegeValueA"
(ByVal lpSystemName$, ByVal lpName$, lpLuid As LUID)

' 指定したアクセストークン内の特権を有効または無効に設定
Declare Function Api_AdjustTokenPrivileges& Lib "advapi32" Alias
"AdjustTokenPrivileges" (ByVal TokenHandle&, ByVal DisableAllPrivileges&, NewState As
TOKEN_PRIVILEGES, ByVal BufferLength&, PreviousState As TOKEN_PRIVILEGES,
ReturnLength&)

'=====================================
'=
'=====================================
Declare Function IsWinNT() As Integer
Function IsWinNT() As Integer
    Var myOS As OSVERSIONINFO
    Var Ret As Long

    myOS.dwOSVersionInfoSize = Len(myOS)
    Ret = Api_GetVersionEx(myOS)
    IsWinNT = (myOS.dwPlatformId = VER_PLATFORM_WIN32_NT)
End Function

'=====================================
'=
'=====================================
Declare Sub EnableShutDown()
Sub EnableShutDown()
    Var hProc As Long
    Var hToken As Long
    Var mLUID As LUID
```

```
    Var mPriv As TOKEN_PRIVILEGES
    Var mNewPriv As TOKEN_PRIVILEGES
    Var Ret As Long

    hProc = Api_GetCurrentProcess()
    Ret = Api_OpenProcessToken(hProc, TOKEN_ADJUST_PRIVILEGES + TOKEN_QUERY, hToken)
    Ret = Api_LookupPrivilegeValue("", "SeShutdownPrivilege", mLUID)

    mPriv.PrivilegeCount = 1
    mPriv.Privileges(0).Attributes = SE_PRIVILEGE_ENABLED
    mPriv.Privileges(0).pLuid = mLUID

    '
    Ret = Api_AdjustTokenPrivileges(hToken, False, mPriv, 4 + (12 *
mPriv.PrivilegeCount), mNewPriv, 4 + (12 * mNewPriv.PrivilegeCount))
End Sub

'=====================================
'=
'=====================================
Declare Sub ShutDownNT(Force As Integer)
Sub ShutDownNT(Force As Integer)
    Var Flags As Long
    Var Ret As Long

    Flags = EWX_SHUTDOWN
    If Force Then Flags = Flags + EWX_FORCE
    If IsWinNT Then EnableShutDown
    Ret = Api_ExitWindowsEx(Flags, 0)
End Sub

'=====================================
'=
'=====================================
Declare Sub RebootNT(Force As Integer)
Sub RebootNT(Force As Integer)
    Var Flags As Long
    Var Ret As Long

    Flags = EWX_REBOOT
    If Force Then Flags = Flags + EWX_FORCE
    If IsWinNT Then EnableShutDown
    Ret = Api_ExitWindowsEx(Flags, 0)
End Sub

'=====================================
'=
'=====================================
Declare Sub LogOffNT(Force As Integer)
Sub LogOffNT(Force As Integer)
    Var Flags As Long
    Var Ret As Long

    Flags = EWX_LOGOFF
    If Force Then Flags = Flags + EWX_FORCE
    Ret = Api_ExitWindowsEx(Flags, 0)
End Sub


'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    A% = MessageBox(GetWindowText, "ログオフしますか？", 4, 1)
    If A% <> 5 Then Exit Sub
    LogOffNT True
End Sub
```

```
'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    A% = MessageBox(GetWindowText, "再起動しますか？", 4, 1)
    If A% <> 5 Then Exit Sub
    RebootNT True
End Sub

'=====================================
'=
'=====================================
Declare Sub Button3_on edecl ()
Sub Button3_on()
    A% = MessageBox(GetWindowText, "シャットダウンしますか？", 4, 1)
    If A% <> 5 Then Exit Sub
    ShutDownNT True
End Sub

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

WinHelpの位置･サイズ指定

WinHelpの表示位置とサイズを指定します。
WinHelp WinHelpを呼び出す

例では、表示位置X,Y(左上)と幅、高さを指定してWinHelpを開きます。


```
'=====================================
'= WinHelpの位置･サイズ指定
'=    (WinHelp2.bas)
'=====================================
#include "Windows.bi"

Type INFOWINHELP
    lStructurSize     As Long
    lPosX             As Long
    lPosY             As Long
    lWidthX           As Long
    lHeightY          As Long
    lMaximum          As Long
    sWinName          As String * 2
End Type

' ヘルプファイルを呼び出す
Declare Function Api_WinHelp& Lib "user32" Alias "WinHelpA" (ByVal hWnd&, ByVal
lpHelpFile$, ByVal wCommand&, dwData As Any)

#define HELP_CONTEXT &H1                    '特定ページ
#define HELP_QUIT &H2                       'ファイルを閉じる
#define HELP_INDEX &H3                      '目次ページ
```

```
#define HELP_HELPONHELP &H4                    'ヘルプ使い方
#define HELP_SETINDEX &H5                      'カレントインデックス
#define HELP_FINDER &HB                        '目次ページ
#define HELP_KEY &H101                         'キーワードに一致したヘルプを表示
#define HELP_PARTIALKEY &H105                  '目次トピック呼出
#define HELP_SETWINPOS &H203

Var Shared HelpFileName As String

Var Shared Edit(3) As Object
Var Shared Text(3) As Object
Var Shared Button(1) As Object

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

'=======================================
'=
'=======================================
Declare Sub ShowHelp(ByVal hWnd As Long, HelpFile As String, x As Long, y As Long, xWidth As
Long, yHeight As Long)
Sub ShowHelp(ByVal hWnd As Long, HelpFile As String, x As Long, y As Long, xWidth As Long,
yHeight As Long)
    Var tpWinHelp As INFOWINHELP
    Var Ret As Long
    HelpFile = HelpFileName

    tpWinHelp.lStructurSize = Len(tpWinHelp)
    tpWinHelp.lPosX = x                        'x軸
    tpWinHelp.lPosY = y                        'y軸
    tpWinHelp.lWidthX = xWidth                 '幅
    tpWinHelp.lHeightY = yHeight               '高さ
    tpWinHelp.lMaximum = 1

    Ret = Api_WinHelp(hWnd, HelpFile, HELP_SETWINPOS, tpWinHelp)
End Sub

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

    Ret = Api_WinHelp(GethWnd, HelpFileName, HELP_QUIT, ByVal 0)
End Sub

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var x As Long
    Var y As Long
    Var w As Long
    Var h As Long

    x = Val(Edit(0).GetWindowText)
    y = Val(Edit(1).GetWindowText)
    w = Val(Edit(2).GetWindowText)
    h = Val(Edit(3).GetWindowText)

    ShowHelp GethWnd, "", x, y, w, h
End Sub
```

```
'====================================
'=
'====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    HelpClose
End Sub

'====================================
'=
'====================================
Declare Sub MainForm_QueryClose(Cancel%, Mode%)
Sub MainForm_QueryClose(Cancel%, Mode%)
    HelpClose
    End
End Sub

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

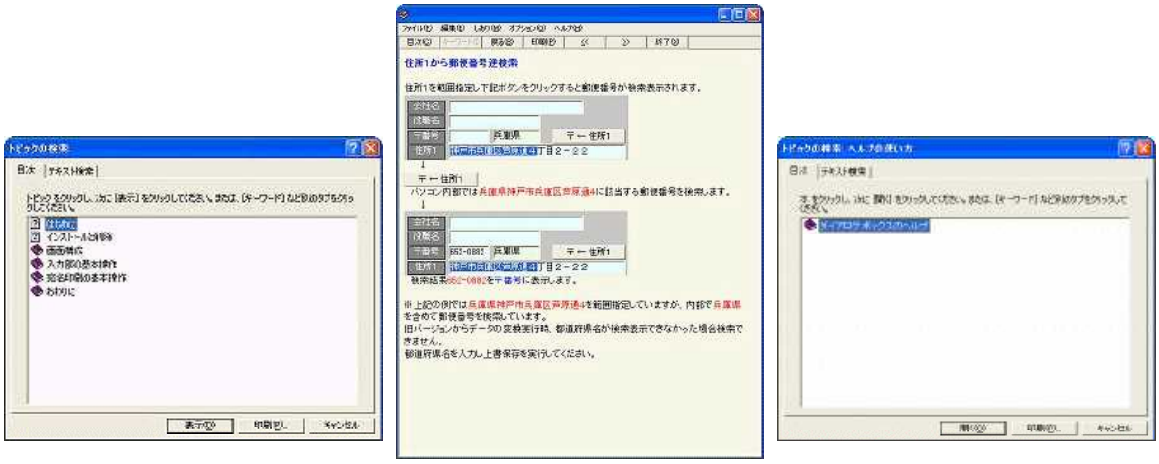
WinHelpを開く

少々古くなった仕様のWinHelpですが目次ページ、指定ページで開きます。
WinHelp WinHelpを使う

★Windows Vista 以降での .hlp の表示(Windows 10 ではサポートされておりません)
Windows Vista 以降では Winhlp32.exe は含まれておらず、.hlp ファイルは表示できませんでしたが、下記URLからダウンロードできます。

- **Windows 8.1 用** Windows Help プログラム (WinHlp32.exe)
  http://www.microsoft.com/ja-jp/download/details.aspx?id=40899
- **Windows 8 用** Windows Help プログラム (WinHlp32.exe)
  https://www.microsoft.com/ja-jp/download/details.aspx?id=35449
- **Windows 7 用** Windows Help プログラム (WinHlp32.exe)
  http://www.microsoft.com/ja-jp/download/details.aspx?id=91
- **Windows Server 2008 用** Windows ヘルプ プログラム (WinHlp32.exe)
  http://www.microsoft.com/ja-jp/download/details.aspx?id=19771
- **Windows Server 2008 R2 用** Windows Help プログラム (WinHlp32.exe)
  http://www.microsoft.com/ja-jp/download/details.aspx?id=4424
- **Windows Vista 用** Windows ヘルプ プログラム (WinHlp32.exe)

左:目次ページで開く「Api_WinHelp(hWnd&, HELPNAME$, HELP_FINDER, ByVal 0)」
中:指定するページで開く「Api_WinHelp(hWnd&, HELPNAME$, HELP_CONTEXT, ByVal 32)」
右:ヘルプの使い方ヘルプを開く

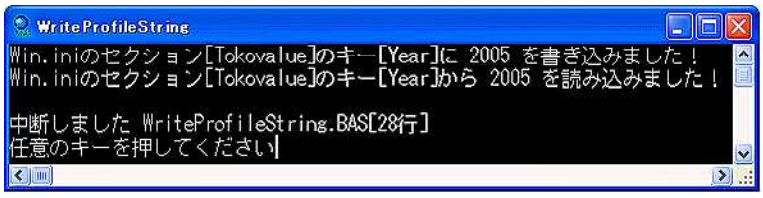


```
'====================================
'= WinHelpを開く
'=    (WinHelp.bas)
'====================================
#include "Windows.bi"

' ヘルプファイルを呼び出す
Declare Function Api_WinHelp& Lib "user32" Alias "WinHelpA" (ByVal hWnd&, ByVal
lpHelpFile$, ByVal wCommand&, ByVal dwData&)

#define HELP_CONTEXT &H1                    '特定ページ
#define HELP_QUIT &H2                       'ファイルを閉じる
#define HELP_INDEX &H3                      '目次ページ
#define HELP_HELPONHELP &H4                 'ヘルプ使い方
#define HELP_SETINDEX &H5                   'カレントインデックス
#define HELP_FINDER &HB                     '目次ページ
#define HELP_KEY &H101                      'キーワードに一致したヘルプを表示
#define HELP_PARTIALKEY &H105               '目次トピック呼出

Var Shared HelpName As String               'ヘルプファイルネーム

'====================================
'=
'====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    HelpName = "AddPrtV62W.hlp"
End Sub

'====================================
'= ヘルプの使い方ヘルプを開く(コンテキストID:0)
'====================================
Declare Sub mnuReadMe_on edecl ()
Sub mnuReadMe_on()
    Var Ret As Long

    Ret = Api_WinHelp(GethWnd, "", HELP_HELPONHELP, ByVal 0)
End Sub

'====================================
'=トピックの検索
'====================================
Declare Sub mnuTopic_on edecl ()
Sub mnuTopic_on()
    Var Ret As Long

    Ret = Api_WinHelp(GethWnd, HelpName, HELP_FINDER, ByVal 0)
End Sub


'====================================
'= 指定するページで開く(コンテキストID:32)
'====================================
```

```
Declare Sub mnuShitei_on edecl ()
Sub mnuShitei_on()
    Var Ret As Long

    Ret = Api_WinHelp(GethWnd, HelpName, HELP_CONTEXT, ByVal 32)
End Sub

'====================================
'= 終了(ヘルプを閉じる)
'====================================
Declare Sub mnuExit_on edecl ()
Sub mnuExit_on()
    Var Ret As Long

    Ret = Api_WinHelp(GethWnd, HelpName, HELP_QUIT, ByVal 0)
    End
End Sub

'====================================
'= 終了(ヘルプを閉じる)
'====================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
    Var Ret As Long

    Ret = Api_WinHelp(GethWnd, HelpName, HELP_QUIT, ByVal 0)
    End
End Sub

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

WIN.INIへの書き込みと読み込み

WIN.INIファイルの指定したセクションおよびキーに文字列を書き込みます。
WriteProfileString WIN.INIの指定のセクションの内容を変更
GetProfileInt WIN.INIから指定のキーの整数値を取得

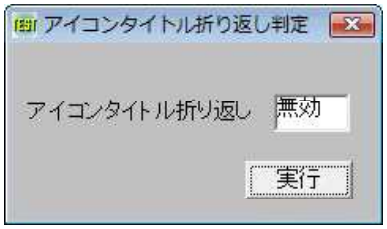


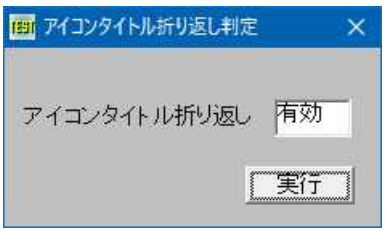

```
'====================================
'= WIN.INIへの書き込みと読み出し
'=    (WriteProfileString.bas)
'====================================

' 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$)

' WIN.INIから指定のキーの整数値を取得
Declare Function Api_GetProfileInt& Lib "kernel32" Alias "GetProfileIntA" (ByVal
lpAppName$, ByVal lpKeyName$, ByVal nDefault&)

var lSec As String
var lKey As String
```

```
var lStr As String
var Ret As Long

lSec = "Tokovalue"
lKey = "Year"
lStr = "2005"

'WIN.INIのセクション「Tokovalue」、キー「Year」に「2005」を書き込む
Ret = Api_WriteProfileString(lSec, lKey, lStr)
Print "Win.iniの" & "セクション[" & lSec & "]のキー[" & lKey & "]に 2005 を書き込みました！"

'WIN.INIのセクション「Tokovalue」、キー「Year」の値を読み込む
Ret = Api_GetProfileInt(lSec, lKey, 0)
Print "Win.iniの" & "セクション[" & lSec & "]のキー[" & lKey & "]から " & Trim$(Str$(Ret)) & " を
読み込みました！"

Stop
End
```

アイコンタイトルの折り返し判定

SystemParametersInfo システム全体に関するパラメータを取得・設定
SPI_GETICONTITLEWRAP(25) アイコンタイトルの折り返しが可能かどうかを調べる



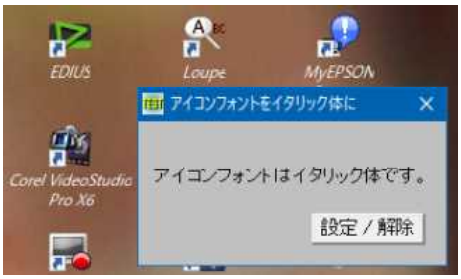



```
'=====================================
'= アイコンタイトルの折り返し判定
'=    (SPI_GETICONTITLEWRAP.bas)
'=====================================
#include "Windows.bi"

#define SPI_GETICONTITLEWRAP 25              'アイコンタイトルの折り返しが可能かどうかを調べる

' 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 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 Wrapping As Long
    Var Ret As Long

    'アイコンの表示要素を取得
    Ret = Api_SystemParametersInfo(SPI_GETICONTITLEWRAP, 0, Wrapping, 0)

    'アイコンの表示要素を表示
    If Wrapping Then
        Text2.SetWindowtext "有効"
```

```
    Else
        Text2.SetWindowtext "無効"
    End If
End Sub

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

アイコンタイトルフォントをイタリック体に

SystemParametersInfo システム全体に関するパラメータを取得・設定
SPI_GETICONMETRICS(45) アイコンに関する寸法情報を定義するICONMETRICS構造体を取得
SPI_SETICONMETRICS(46) アイコンに関する寸法情報を設定
SPIF_UPDATEINIFILE(&H1) 新しい設定をユーザープロファイルに書き込む
SPIF_SENDWININICHANGE(&H2) 実行中のプログラムに更新メッセージを送る
SPIF_SENDCHANGE(&H2) 実行中のプログラムに更新メッセージを送る(SPIF_SENDWININICHANGE)

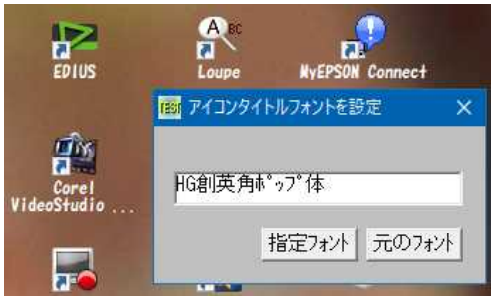



```
'=====================================
'= アイコンタイトルフォントをイタリック体に
'    (IconMetrics.bas)
'=====================================
#include "Windows.bi"

#define LF_FACESIZE 32

Type LOGFONT
    lfHeight          As Long              '文字セルまたは文字の高さ
    lfWidth           As Long              '平均文字幅
    lfEscapement      As Long              '文字送りの方向とX軸との角度
    lfOrientation     As Long              'ベースラインとX軸との角度
    lfWeight          As Long              'フォントの太さ
    lfItalic          As Byte              'イタリック体指定
    lfUnderline       As Byte              '下線付き指定
    lfStrikeOut       As Byte              '打ち消し線付き指定
    lfCharSet         As Byte              'キャラクタセット
    lfOutPrecision    As Byte              '出力精度
    lfClipPrecision   As Byte              'クリッピングの精度
    lfQuality         As Byte              '出力品質
    lfPitchAndFamily  As Byte              'ピッチとファミリ
    lfFaceName(LF_FACESIZE - 1) As Byte    'フォント名
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構造体を取得
#define SPI_SETICONMETRICS 46                   'アイコンに関する寸法情報を設定
#define SPIF_UPDATEINIFILE &H1                  '新しい設定をユーザープロファイルに書き込む
#define SPIF_SENDWININICHANGE &H2               '実行中のプログラムに更新メッセージを送る
#define SPIF_SENDCHANGE &H2                     '実行中のプログラムに更新メッセージを送る(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 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 im As ICONMETRICS
    Var Ret As Long

    'アイコンの表示要素を定義する構造体を初期化
    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指定
    im.lfFont.lfItalic = im.lfFont.lfItalic Xor 1

    'アイコンの表示要素を設定
    Ret = Api_SystemParametersInfo(SPI_SETICONMETRICS, Len(im), im, SPIF_UPDATEINIFILE
Or SPIF_SENDCHANGE)

    If im.lfFont.lfItalic = 0 Then
        Text1.SetWindowText "アイコンフォントは標準です。"
    Else
        Text1.SetWindowText "アイコンフォントはイタリック体です。"
    End If
End Sub

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

アイコンタイトルフォントを設定

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
SPI_GETICONTITLELOGFONT(31) アイコンのタイトルに使われるフォントのLOGFONT構造体を取得
SPI_SETICONTITLELOGFONT(34) アイコンのタイトルに使われるフォントを設定
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




```
'====================================
'= アイコンタイトルフォントを設定
'=     (SetIconTitleFont.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

#define SPI_GETICONTITLELOGFONT 31          'アイコンのタイトルに使われるフォントのLOGFONT構造体を取得
#define SPI_SETICONTITLELOGFONT 34          'アイコンのタイトルに使われるフォントを設定
#define SPIF_SENDWININICHANGE &H2           '全てのアプリケーションに通知して更新する
#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&)

' 文字列をコピーする
Declare Function Api_lstrcpy$ Lib "Kernel32" Alias "lstrcpyA" (lpszString1 As Any, ByVal
lpszString2$)

Var Shared OriginalFont As LOGFONT

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 MainForm_Start edecl ()
Sub MainForm_Start()
    Var Ret As Long

    Text1.SetWindowText "HG創英角ﾎﾟｯﾌﾟ体"
```

```
    Ret = Api_SystemParametersInfo(SPI_GETICONTITLELOGFONT, Len(OriginalFont),
OriginalFont, 0)
End Sub

'====================================
'=
'====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var lf As LOGFONT
    Var LogFont As String * 255
    Var Ret As Long

    'アイコンの表示要素を取得
    Ret = Api_SystemParametersInfo(SPI_GETICONTITLELOGFONT, Len(lf), lf, 0)

    'アイコンの表示要素を表示
    Text1.SetWindowText "HG創英角ﾎﾟｯﾌﾟ体"

    'アイコンのフォント名を指定
    LogFont = "HG創英角ﾎﾟｯﾌﾟ体" & Chr$(0)
    LogFont = Api_lstrcpy(lf.lfFaceName(0), LogFont)

    'アイコンの表示要素を設定
    Ret = Api_SystemParametersInfo(SPI_SETICONTITLELOGFONT, Len(lf), lf,
SPIF_UPDATEINIFILE Or SPIF_SENDWININICHANGE)
End Sub

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

    Text1.SetWindowText "MS UI Gothic"

    'アイコンの表示要素を設定
    Ret = Api_SystemParametersInfo(SPI_SETICONTITLELOGFONT, Len(OriginalFont),
OriginalFont, SPIF_UPDATEINIFILE Or SPIF_SENDWININICHANGE)
End Sub

'====================================
'=
'====================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
    Button2_on
End Sub

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

アイコンをカーソルに

アイコンをカーソルにします。
CopyCursor カーソルのコピーを作成する
LoadImage 画像ファイルの読み込み
SetCursor マウスカーソルを設定する
DestroyIcon アイコンのハンドルを破棄する


```
'=====================================
'= アイコンをカーソルに (IconToCursor.bas)
'=====================================
#include "Windows.bi"

' カーソルのコピーを作成する
Declare Function Api_CopyCursor& Lib "user32" Alias "CopyIcon" (ByVal pcur&)

' 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&)

' マウスカーソルを設定する
Declare Function Api_SetCursor& Lib "user32" Alias "SetCursor" (ByVal hCursor&)

' カーソルを破棄する
Declare Function Api_DestroyCursor& Lib "user32" Alias "DestroyCursor" (ByVal hCursor&)

' アイコンのハンドルを破棄する
Declare Function Api_DestroyIcon& Lib "user32" Alias "DestroyIcon" (ByVal hIcon&)

#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

Var Shared hCursor As Long
Var Shared hOldCursor As Long

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

    'アイコンファイルからカーソルを作る
    hIcon = Api_LoadImage(GethInst, "Pencil.ico", IMAGE_ICON, 0, 0, LR_LOADFROMFILE)

    '読み込んだアイコンをカーソルとしてコピー
    hCursor = Api_CopyCursor(hIcon)

    '元のカーソルを記憶
    hOldCursor = Api_SetCursor(hCursor)
    Ret = Api_DestroyIcon(hIcon)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_MouseMove edecl(ByVal Button%, ByVal Shift%, ByVal SX!, ByVal SY!)
Sub MainForm_MouseMove(ByVal Button%, ByVal Shift%, ByVal SX!, ByVal SY!)
    Var Ret As Long

    'カーソルが移動するたびに元の形状に戻ろうとするので再設定する
    Ret = Api_SetCursor(hCursor)
End Sub
```

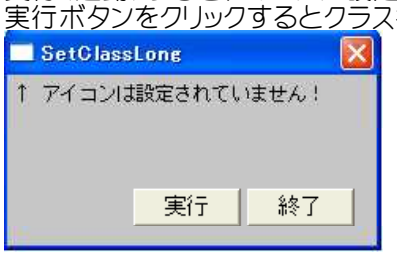
```
'=====================================
'=
'=====================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
    Var Ret As Long

    '元のカーソルに戻す
    Ret = Api_SetCursor(hOldCursor)
    Ret = Api_DestroyCursor(hCursor)
End Sub

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

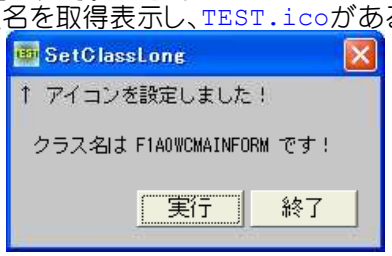
アイコンを設定する

アイコンを設定します。
GetClassLong クラスに関連付けてる補足データ域からlong値を取得
SetClassLong クラスに関連付けている補足データ域にlong値を設定
GetClassName ウィンドウクラス名を取得
CallWindowProc 指定されたウィンドウプロシージャにメッセージ情報を渡す関数
LoadImage 画像ファイルの読み込み

例では、テストプログラムと同じフォルダにTEST.icoを用意していますが、インポートしていない状態です。

実行（起動）するとアイコンは設定されていません。
実行ボタンをクリックするとクラス名を取得表示し、TEST.icoがある場合メニューバーにアイコンをセットします。





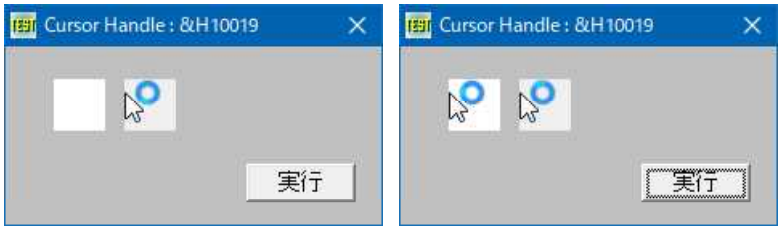

```
'=====================================
'= アイコンを設定する
'=    (SetClassLong.bas)
'=====================================
#include "Windows.bi"

' クラスに関連付けてる補足データ域から Long 値を取得
Declare Function Api_GetClassLong& Lib "user32" Alias "GetClassLongA" (ByVal hWnd&,
ByVal nIndex&)

' 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&)

' ウィンドウクラス名を取得
Declare Function Api_GetClassName& Lib "user32" Alias "GetClassNameA" (ByVal hWnd&,
```

```
ByVal lpClassName$, ByVal nMaxCount&)

' 指定されたウィンドウプロシージャにメッセージ情報を渡す関数
Declare Function Api_CallWindowProc& Lib "user32" Alias "CallWindowProcA" (ByVal
lpPrevWndFunc&, ByVal hWnd&, ByVal Msg&, ByVal WParam&, ByVal lParam&)

' 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 GCL_CBCLSEXTRA -20                'クラスに関連付けられている拡張クラスメモリのサイズをバイト単位で設定
#define GCL_CBWNDEXTRA -18                'ウィンドウに関連付けられている拡張ウィンドウメモリのサイズをバイト単位で設定
#define GCL_HBRBACKGROUND -10             'クラスに関連付けられている背景ブラシのハンドルを書き換える
#define GCL_HCURSOR -12                   'クラスに関連付けられているマウスカーソルのハンドルを書き換える
#define GCL_HICON -14                     'クラスに関連付けられているアイコンのハンドルを書き換える
#define GCL_HMODULE -16                   'クラスを登録したモジュールのハンドルを書き換える
#define GCL_MENUNAME -8                   'メニュー名が入った文字列のアドレスを書き換える
#define GCL_STYLE -26                     'ウィンドウクラスのスタイルビットを書き換える
#define GCL_WNDPROC -24                   'クラスに関連付けられているウィンドウプロシージャのアドレスを書き換える


#define IMAGE_BITMAP 0                    'ビットマップ
#define IMAGE_ICON 1                      'アイコン
#define IMAGE_CURSOR 2                    'カーソル
#define IMAGE_ENHMETAFILE 3               '拡張メタファイル

#define LR_CREATEDIBSECTION &H2000        'デバイス独立ビットマップとしてロードする
#define LR_DEFAULTCOLOR &H0               'デフォルト
#define LR_DEFAULTSIZE &H40               '幅・高さの指定がゼロであれば、システムメトリック値のサイズを採用する
#define LR_LOADFROMFILE &H10              '外部ファイルからロードする
#define LR_LOADMAP3DCOLORS &H1000         'カラーテーブルを走査して、3Dカラーをシステム値に置き換える
#define LR_LOADTRANSPARENT &H20           '最初のピクセルデータを背景色と見なし、システム値に置き換える
#define LR_MONOCHROME &H1                 '白黒イメージとしてロード
#define LR_SHARED &H8000                  'イメージハンドルを固定する
#define LR_COPYFROMSOURCE &H4000          'リソースから読み込む
#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 hIcon As Long

Var Shared Text1 As Object
Var Shared Text2 As Object
Var Shared Button1 As Object
Var Shared Button2 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 12
Text2.Attach GetDlgItem("Text2") : Text2.SetFontSize 12
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14

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

    Text2.SetWindowText "↑ アイコンは設定されていません！"
End Sub

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

```
    Var Buffer As String * 256
    Var Ret As Long

    Ret = Api_GetClassName(GethWnd, Buffer, Len(Buffer))
    If Ret = 0 Then
        Text1.SetWindowText "クラス名を取得できません！"
    Else
        Text1.SetWindowText "クラス名は " & Left$(Buffer, Ret) & " です！"
    End If

    hIcon = Api_LoadImage(GethInst, "TEST.ico", IMAGE_ICON, 32, 32, LR_LOADFROMFILE)
    If hIcon = 0 Then
        Text2.SetWindowText "アイコンはありません！"
        Wait 50
        Exit Sub
    End If

    Ret = Api_SetClassLong(GethWnd, GCL_HICON, hIcon)
    If Ret = 0 Then
        Text2.SetWindowText "アイコンを設定できません！"
    Else
        Text2.SetWindowText "↑ アイコンを設定しました！"
    End If
End Sub

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

    ProcAddress = Api_GetClassLong(GethWnd, GCL_WNDPROC)
    If ProcAddress = 0 Then
        Text1.SetWindowText "ウィンドウクラス名の取得はできません！"
        Wait 50
        End
    End If

    Ret = Api_CallWindowProc(ProcAddress, GethWnd, WM_CLOSE, 0, 0)
End Sub

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

アイコンを描画

DrawIcon アイコンを描画
DrawIconEx アイコンを描画
Sleep カレントスレッドの実行を指定の時間だけ中断
LoadImage 画像ファイルの読み込み
DestroyCursor カーソルを破棄する
SendMessage ウィンドウにメッセージを送信
CreateWindowEx ウィンドウ(コントロール)を作成
DestroyWindow CreateWindowExの解放
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




```
'=====================================
'= アイコンを描画
'=    (DrawIcon2.bas)
'=====================================
#include "Windows.bi"

' 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&)

' 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&)

' 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&)

' カーソルを破棄する
Declare Function Api_DestroyCursor& Lib "user32" Alias "DestroyCursor" (ByVal hCursor&)

' 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&)

' 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 hParent&, ByVal hMenu&, ByVal hInstance&, lpParam As Any)

' CreateWindowExの解放
Declare Function Api_DestroyWindow& Lib "user32" Alias "DestroyWindow" (ByVal hWnd&)

' 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 DI_DEFAULTSIZE 8                    'widthとheightが0である場合、アイコン(マウスカーソル)をデフォルトのサイズで描画

#define DI_NORMAL 3                         'DI_IMAGEとDI_MASKの組み合わせ
#define IMAGE_CURSOR 2                      'カーソル
#define LR_DEFAULTSIZE &H40                 '標準サイズで表示
#define LR_SHARED &H8000                    'イメージハンドルを固定する
#define OCR_APPSTARTING 32650               '標準の矢印カーソルと小さい砂時計カーソル
#define OCR_NORMAL 32512                    'Windows組込ビットマップ
#define SS_ICON &H3                         'アイコンを表示するスタティックコントロールを作成
#define STM_SETIMAGE &H172                  'ピクチャーコントロールのビットマップを変更
#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
#define WS_VISIBLE &H10000000               '可視状態のウィンドウを作成する

Var Shared Picture1 As Object
Var Shared Picture2 As Object
Var Shared Button1 As Object

Picture1.Attach GetDlgItem("Picture1")
```

```
Picture2.Attach GetDlgItem("Picture2")
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var Shared hWndCur As Long
Var Shared hCur As Long
Var Shared hDC As Long

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

    hDC = Api_GetDC(Picture1.GethWnd)

    hCur = Api_LoadImage(ByVal 0, OCR_APPSTARTING, IMAGE_CURSOR, 0, 0, LR_SHARED Or
LR_DEFAULTSIZE)

    SetWindowText "Cursor Handle : &H" & Hex$(hCur)

    hWndCur = Api_CreateWindowEx(0, "Static", ByVal 0, WS_CHILD Or WS_VISIBLE Or SS_ICON,
0, 0, 32, 32, Picture2.GethWnd, ByVal 0, GethInst, ByVal 0)

    Ret = Api_SendMessage(hWndCur, STM_SETIMAGE, IMAGE_CURSOR, hCur)
End Sub

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

    i = 0
    While 0 <> Api_DrawIconEx(hDC, 0, 0, hCur, 0, 0, I, 0, DI_DEFAULTSIZE Or DI_NORMAL)
        i = i + 1
        CallEvent
        Api_Sleep(100)
        Picture1.Cls
    Wend
    Ret = Api_DrawIconEx(hDC, 0, 0, hCur, 0, 0, 0, 0, DI_DEFAULTSIZE Or DI_NORMAL)
End Sub

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

    Ret = Api_DestroyWindow(hWndCur)
    Ret = Api_ReleaseDC(hDC, Picture1.GethWnd)
End Sub

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

アイドル時間を取得

アイドル時間を取得します。
GetTickCount システムが起動してからの経過時間を取得
GetLastInputInfo 最後に発生した入力イベントの時刻を取得

例では、カーソルの動きを関知して時間をリセットしています。

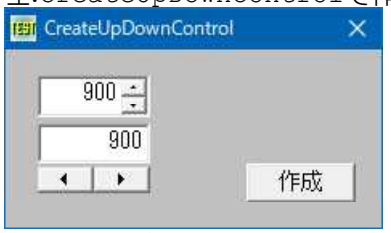

```
'=====================================
'= アイドル時間を取得
'= (GetLastInputInfo.bas)
'=====================================
#include "Windows.bi"

Type LASTINPUTINFO
   cbSize      As Long
   dwTime      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" ()

' 最後に発生した入力イベントの時刻を取得
Declare Function Api_GetLastInputInfo& Lib "user32" Alias "GetLastInputInfo" (plii As
Any)

Var Shared Timer1 As Object
Var Shared Text1 As Object
Var Shared Button1 As Object

Timer1.Attach GetDlgItem("Timer1")
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var Shared Flg As Integer

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
   Timer1.SetInterval 10
   Timer1.Enable -1
   Text1.SetWindowText "アイドル時間(秒)"

   Button1.SetWindowText "Stop"
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Flg = Not Flg

    Timer1.Enable Flg

    Select Case Flg
        Case -1
            Button1.SetWindowText "Stop"
        Case 0
```

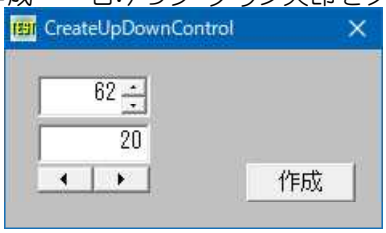
```
            Button1.SetWindowText "Start"
    End Select
End Sub

'=======================================
'=
'=======================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var lii As LASTINPUTINFO
    Var Ret As Long

    lii.cbSize = Len(lii)
    Ret = Api_GetLastInputInfo(lii)

    Text1.SetWindowText "アイドル時間(秒)" & Chr$(13, 10) & Str$((Api_GetTickCount() -
lii.dwTime) / 1000)
End Sub

'=======================================
'=
'=======================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
   Timer1.Enable 0
End Sub

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

アップダウンコントロールの作成(Ⅰ)

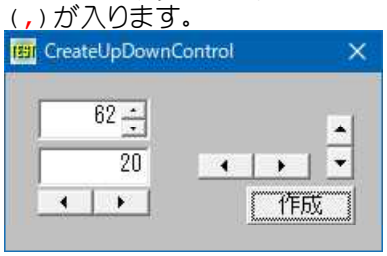
アップダウンコントロールを作成します。CreateUpDownControlおよびCreateWindowExでも作成できますが、前者のほうが簡単です。
CreateUpDownControl アップダウンコントロールを作成
CreateWindowEx ウインドウ(コントロール)を作成
DestroyWindow CreateWindowExの解放

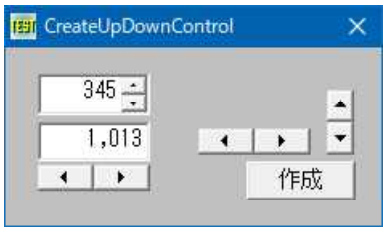
左:CreateUpDownControlで作成　　右:アップ・ダウン矢印をクリックした状態



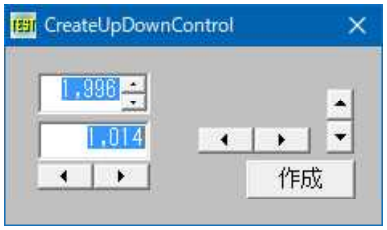


左:「作成」ボタンをクリックし、CreateWindowExで作成　　右:nMaxを1000以上に設定した場合、3桁毎にカンマ(,)が入ります。





コードの中でコメント行のようにor UDS_NOTHOUSANDSを追加した場合(Edit2の表示は、3桁毎のカンマは入りません)

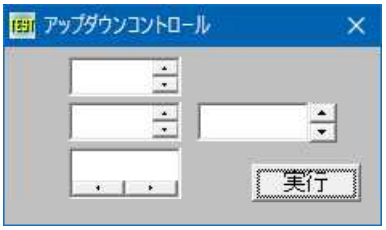



```
'=====================================
'= アップダウンコントロールの作成
'=    (CreateUpDownControl.bas)
'=====================================
#include "Windows.bi"

#define WS_VISIBLE &H10000000              '表示する
#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 PBM_SETSTEP &H404                  '(WM_USER + 4)増分の設定
#define UPDOWN_CLASS "msctls_updown32"     'アップダウンコントロール

#define ICC_ANIMATE_CLASS &H80             'アニメートコントロール
#define ICC_BAR_CLASSES &H4                'ツールバー、ステータスバー、トラックバー、ツールチップ
#define ICC_COOL_CLASSES &H400             'レバーコントロール
#define ICC_DATE_CLASSES &H100             'DTPコントロール
#define ICC_HOTKEY_CLASS &H40              'ホットキーコントロール
#define ICC_INTERNET_CLASSES &H800         'IPアドレスコントロール(Version4.71以降)
#define ICC_LISTVIEW_CLASSES &H1           'リストビュー、ヘッダーコントロール
#define ICC_NATIVEFNTCTL_CLASS &H2000      'ネィティブフォントコントロール
#define ICC_PAGESCROLLER_CLASS &H1000      'ページャーコントロール(Version4.71以降)
#define ICC_PROGRESS_CLASS &H20            'プログレスバー
#define ICC_TAB_CLASSES &H8                'タブコントロール、ツールチップ
#define ICC_TREEVIEW_CLASSES &H2           'ツリービュー、ツールチップ
#define ICC_UPDOWN_CLASS &H10              'アップダウンコントロール
#define ICC_USEREX_CLASSES &H200           '拡張コンボボックス
#define ICC_WIN95_CLASSES &HFF             'すべてのコントロール

#define UDM_GETPOS &H468                   '現在のポジションを取得
#define UDM_GETPOS32 &H472                 '現在のポジション(32ビット値)を取得
#define UDM_SETBUDDY &H469                 'バディウィンドウを設定
#define UDM_SETPOS &H467                   '現在のポジション(32ビット値)を設定
#define UDM_SETPOS32 &H471                 '現在のポジションを設定
#define UDM_SETRANGE &H465                 'ポジションの範囲を設定
#define UDM_SETRANGE32 &H46F               'ポジションの範囲(32ビット値)を設定

#define UDS_ALIGNLEFT &H8                  'アップダウンをバディウィンドウの左に配置
#define UDS_ALIGNRIGHT &H4                 'アップダウンをバディウィンドウの右に配置
#define UDS_ARROWKEYS &H20                 '矢印キーで現在位置を変更できる(スピンコントロールの場合)
#define UDS_AUTOBUDDY &H10                 'Zオーダが一つ前のウィンドウを自動的にバディとする
#define UDS_HORZ &H40                      '水平アップダウンにする
#define UDS_NOTHOUSANDS &H80               '三桁ごとの区切り記号を表示しない(スピンコントロールの場合)
#define UDS_SETBUDDYINT &H2                '位置の変更をバディウィンドウに自動的に表示する
#define UDS_WRAP &H1                       '範囲を越えたとき値を折り返す

' アップダウンコントロールを作成
Declare Function Api_CreateUpDownControl& Lib "comctl32" Alias "CreateUpDownControl"
(ByVal dwStyle&, ByVal x&, ByVal y&, ByVal nWidth&, ByVal nHeight&, ByVal hParent&, ByVal
ID&, ByVal hInstance&, ByVal hBuddy&, ByVal max&, ByVal min&, ByVal StartPos&)

' ウインドウ(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&)

' CreateWindowExの解放
Declare Function Api_DestroyWindow& Lib "user32" Alias "DestroyWindow" (ByVal hWnd&)

Var Shared Edit1 As Object
```

```
Var Shared Edit2 As Object
Var Shared Button1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Edit2.Attach GetDlgItem("Edit2") : Edit2.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var Shared hUpDownCtrl As Long

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var nMin As Long
    Var nMax As Long
    Var nPos As Long
    Var Ret As Long

    nMin = 0
    nMax = 2000
    nPos = 900

    'アップダウンコントロールを作成する
    Ret = Api_CreateUpDownControl(WS_CHILD Or WS_BORDER Or WS_VISIBLE Or UDS_SETBUDDYINT
Or UDS_ALIGNRIGHT, 0, 0, 0, 0, GethWnd, 100, GetInst, Edit1.GethWnd, nMax, nMin, nPos)
    Ret = Api_CreateUpDownControl(WS_CHILD Or WS_BORDER Or WS_VISIBLE Or UDS_SETBUDDYINT
Or UDS_HORZ, 20, 72, 70, 16, GethWnd, 100, GetInst, Edit2.GethWnd, nMax, nMin, nPos)

'   Ret = Api_CreateUpDownControl(WS_CHILD Or WS_BORDER Or WS_VISIBLE Or UDS_SETBUDDYINT
Or UDS_HORZ Or UDS_NOTHOUSANDS, 20, 72, 70, 16, GethWnd, 100, GetInst, Edit2.GethWnd,
nMax, nMin, nPos)

End Sub

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

    hUpDownCtrl = Api_CreateWindowEx(0, UPDOWN_CLASS, "", WS_CHILD Or WS_VISIBLE Or
ICC_UPDOWN_CLASS, 200, 26, 0, 40, GethWnd, 0, 0, 0)
    hUpDownCtrl = Api_CreateWindowEx(0, UPDOWN_CLASS, "", WS_CHILD Or WS_VISIBLE Or
ICC_UPDOWN_CLASS Or UDS_HORZ, 120, 50, 70, 16, GethWnd, 0, 0, 0)
End Sub

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

    Ret = Api_DestroyWindow(hUpDownCtrl)
End Sub

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

アップダウンコントロールの作成(II)

CreateWindowExでアップダウンコントロールを作成します。
CreateWindowEx ウインドウ(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ウィンドウクラスを登録して初期化
SetParent 指定された子ウィンドウの親ウィンドウを変更

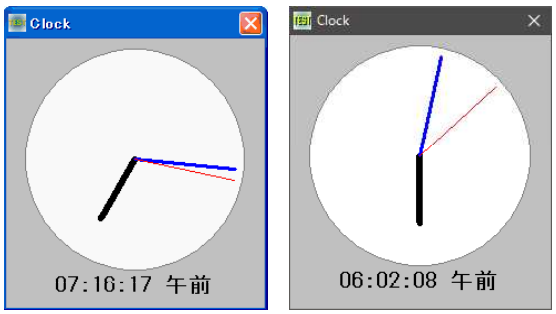



```
'========================================
'= アップダウンコントロールをコードで作成(II)
'=    (CreateWindowEx5.bas)
'========================================
#include "Windows.bi"

#define UPDOWN_CLASS "msctls_updown32"
#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_VISIBLE &H10000000                '可視状態のウィンドウを作成する

#define UDS_AUTOBUDDY &H10                   'Zオーダが一つ前のウィンドウを自動的にバディとする
#define UDS_HORZ &H40                        '水平アップダウンにする

' 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&)

' 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。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)

' 指定された子ウィンドウの親ウィンドウを変更
Declare Function Api_SetParent& Lib "user32" Alias "SetParent" (ByVal hWndChild&, ByVal
hWndNewParent&)
Var Shared Text1 As Object
Var Shared Edit1 As Object
Var Shared Edit2 As Object
Var Shared Edit3 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Edit2.Attach GetDlgItem("Edit2") : Edit2.SetFontSize 14
Edit3.Attach GetDlgItem("Edit3") : Edit3.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Var Shared UDControl As Long

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

```
    Api_InitCommonControls

    'TextBoxに作成
    UDControl = Api_CreateWindowEx(0, UPDOWN_CLASS, ByVal 0, WS_VISIBLE Or WS_CHILD,
Text1.GetWidth - 20, 0, 5, 20, GethWnd, 0, GethInst, ByVal 0)
    Ret = Api_SetParent(UDControl, Text1.GethWnd)

    'EditBoxに作成(垂直)
    UDControl = Api_CreateWindowEx(0, UPDOWN_CLASS, ByVal 0, WS_VISIBLE Or WS_CHILD,
Edit1.GetWidth - 20, 0, 5, 20, GethWnd, 0, GethInst, ByVal 0)
    Ret = Api_SetParent(UDControl, Edit1.GethWnd)

    'EditBoxに作成(水平)
    UDControl = Api_CreateWindowEx(0, UPDOWN_CLASS, ByVal 0, WS_VISIBLE Or WS_CHILD Or
UDS_HORZ, 0, Edit2.GetHeight - 14, 66, 10, GethWnd, 0, GethInst, ByVal 0)
    Ret = Api_SetParent(UDControl, Edit2.GethWnd)

    'Form上の指定した位置に作成
    UDControl = Api_CreateWindowEx(0, UPDOWN_CLASS, ByVal 0, WS_VISIBLE Or WS_CHILD, 190,
35, 10, 24, GethWnd, 0, GethInst, ByVal 0)
End Sub

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

    Ret = Api_DestroyWindow(UDControl)
End Sub

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

アナログ時計

簡易アナログ時計を作成します。
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




```
'=======================================
'= 簡易アナログ時計
'=    (Clock.bas)
'=======================================
#include "Windows.bi"

Type SYSTEMTIME
```

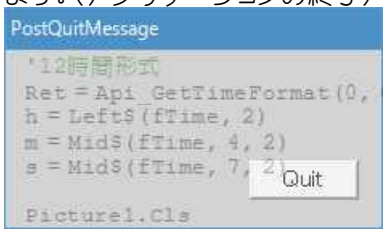
```
    wYear             As Integer
    wMonth            As Integer
    wDayOfWeek        As Integer
    wDay              As Integer
    wHour             As Integer
    wMinute           As Integer
    wSecond           As Integer
    wMilliseconds     As Integer
End Type

 ' スレッドのロケールIDを取得
Declare Function Api_GetThreadLocale& Lib "Kernel32" Alias "GetThreadLocale" ()

' スレッドのロケールID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)

' 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&)

#define LOCALE_SYSTEM_DEFAULT &H400        'システムのデフォルトロケール

Var Shared Timer1 As Object
Var Shared Picture1 As Object
Var Shared Text1 As Object

Timer1.Attach GetDlgItem("Timer1")
Picture1.Attach GetDlgItem("Picture1")
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 20
Var Shared x As Integer
Var Shared y As Integer
Var Shared w As Integer

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    x = Picture1.GetWidth / 2
    y = Picture1.GetHeight / 2
    w = Picture1.GetWidth

    Timer1.SetInterval 50
    Timer1.Enable -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var ST As SYSTEMTIME
    Var fTime As String * 256
    Var h As String
    Var m As String
    Var s As String
    Var Ret As Long

    If Api_GetThreadLocale <> LOCALE_SYSTEM_DEFAULT Then
        Ret = Api_SetThreadLocale(LOCALE_SYSTEM_DEFAULT)
    End If

    Api_GetLocalTime ST
```

```
    '12時間形式
    Ret = Api_GetTimeFormat(0, 0, ST, "hh:mm:ss tt", fTime, Len(fTime))
    h = Left$(fTime, 2)
    m = Mid$(fTime, 4, 2)
    s = Mid$(fTime, 7, 2)

    Picture1.Cls
    Picture1.SetDrawWidth 1
    Picture1.Circle (x, y), w / 2 - 1, 1,,,,f,, 15

    Picture1.SetDrawWidth 6
    Picture1.Line (x, y)-(x + (60 * Sin(Val(h) * 3.1416 / 6)), y - (60 * Cos(Val(h) * 3.1416
/ 6))),, 0

    Picture1.SetDrawWidth 3
    Picture1.Line (x, y)-(x + (90 * Sin(Val(m) * 3.1416 / 30)), y - (90 * Cos(Val(m) * 3.1416
/ 30))),, 3

    Picture1.SetDrawWidth 1
    Picture1.Line (x, y)-(x + (92 * Sin(Val(s) * 3.1416 / 30)), y - (92 * Cos(Val(s) * 3.1416
/ 30))),, 5

    Text1.SetWindowText Left$(fTime, Ret)
End Sub

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

アプリケーション停止キューをWindowsに送る

アプリケーションを停止するキューをWindowsに送ります。
SetLayeredWindowAttributes 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
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
PostQuitMessage アプリケーションを停止するキューをWindowsに送る

例では、ウィンドウ(Form)を透過処理し「Quit」ボタンクリックでアプリケーションを停止するキューをWindowsに送ります。(アプリケーションの終了)

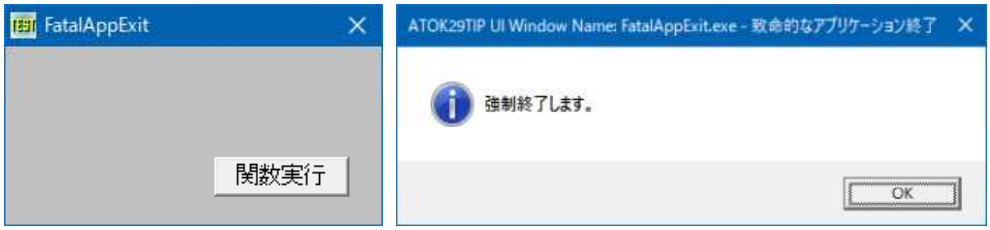



```
'=======================================
'= アプリケーションを停止するキューをWindowsに送る
'=    (PostQuitMessage.bas)
'=======================================
#include "Windows.bi"

' 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&)

' 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&)

' アプリケーションを停止するキューをWindowsに送る
Declare Sub Api_PostQuitMessage Lib "user32" Alias "PostQuitMessage" (ByVal nExitCode&)

#define GWL_EXSTYLE -20                         '拡張ウィンドウスタイル
#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)

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

   Ret = Api_SetWindowLong(GethWnd, GWL_EXSTYLE, Api_GetWindowLong(GethWnd,
GWL_EXSTYLE) Or WS_EX_LAYERED)

   '透過処理
   Ret = Api_SetLayeredWindowAttributes(GethWnd, 0, 180, LWA_ALPHA)
End Sub

'=======================================
'= アプリケーションを停止するキューをWindowsに送る
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()

    Api_PostQuitMessage(0)
End Sub

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

アプリケーションを強制終了

「致命的なアプリケーション終了」というメッセージボックスを表示させ、即座にプログラムを終了させることが出来ます。
FatalAppExit メッセージボックスを表示して、アプリケーションを終了




```
'=======================================
'= アプリケーションを強制終了
'=    (FatalAppExit.bas)
'=======================================
#include "Windows.bi"
```

```
' メッセージボックスを表示して、アプリケーションを終了
Declare Sub Api_FatalAppExit Lib "kernel32" Alias "FatalAppExitA" (ByVal uAction&, ByVal
lpMessageText$)

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var MSG As String

    MSG = "強制終了します。"                    'メッセージボックスに表示する文字列を指定
    Api_FatalAppExit 0, MSG                    'アプリケーションの強制終了を実行

End Sub

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

ある点が指定領域内かどうかの判断

ある点が指定した領域内にあるかどうかを判断します。
PtInRect ある点が指定領域内にあるかどうかを判断
GetCursorPos マウスポインタの現在位置に相当するスクリーン座標を取得

カーソルが設定領域内にある時「範囲内」、設定外にあるとき「範囲外」と、スクリーン座標とともに表示されます。


```
'=====================================
'= 指定した点が指定領域内かどうかの判断
'=    (PtInRect.bas)
'=====================================
#include "Windows.bi"

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

' 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&)

' 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 Text(2) As Object
Var Shared Timer1 As Object

For i = 0 To 2
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1))) : Text(i).SetFontSize 14
Next
Timer1.Attach GetDlgItem("Timer1")

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var txt As String
    Var CrLf As String

    Crlf = Chr$(13, 10)
    txt = txt & "設定範囲" & CrLf
    txt = txt & "rc.Left   = 100" & CrLf
    txt = txt & "rc.Top    = 100" & CrLf
    txt = txt & "rc.Right  = 600" & CrLf
    txt = txt & "rc.Bottom = 400"

    Text(0).SetWindowText txt

    Timer1.SetInterval 10
    Timer1.Enable -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var pt As POINTAPI
    Var rc As RECT
    Var Ret As Long

    rc.Left = 100
    rc.Top = 100
    rc.Right = 600
    rc.Bottom = 400

    Ret = Api_GetCursorPos(pt)
    Text(1).SetWindowText Format$(pt.x, "####") & "/" & Format$(pt.y, "####")

    If Api_PtInRect(rc, pt.x, pt.y) = 1 Then
        Text(2).SetFontBold -1
        Text(2).SetWindowText "範囲内"
    Else
        Text(2).SetFontBold 0
        Text(2).SetWindowText "範囲外"
    End If
End Sub

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

一時停止中のウィンドウ再描画

CreateWaitableTimer「待機可能」タイマオブジェクトを作成

OpenWaitableTimer 既存の名前付き「待機可能」タイマオブジェクトのハンドルを取得
SetWaitableTimer 指定した「待機可能」タイマをアクティブにする
CancelWaitableTimer 指定した「待機可能」タイマをアクティブでない状態に設定
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
MsgWaitForMultipleObjects シグナル状態になったとき、またはタイムアウト時間が経過したとき制御を戻す
Sleep カレントスレッドの実行を指定の時間だけ中断

例では、どちらも約10秒間一時停止させています。
「Sleep」で一時停止中に他のウィンドウが被った場合は再描画されません。「SetWaitableTimer」で一時停止中は再描画されます。

参照
http://support.microsoft.com/kb/231298/ja
Visual BasicでSetWaitableTimerを使用する方法

```
'=======================================
'= 一時停止中のウィンドウ再描画
'=    (SetWaitableTimer.bas)
'=======================================
#include "Windows.bi"

Type FILETIME
    dwLowDateTime     As Long
    dwHighDateTime    As Long
End Type

#define WAIT_ABANDONED &H80                 'オブジェクトがmutexの場合だけ
#define WAIT_ABANDONED_0 &H80               'オブジェクトがシグナル状態
#define WAIT_FAILED -1                      'エラーが発生したことを示す
#define WAIT_IO_COMPLETION &HC0             'I/O完了コールバック関数の呼び出しによって戻った
#define WAIT_OBJECT_0 &H0                   'オブジェクトがシグナル状態になったことを示す
#define WAIT_OBJECT_1 1                     '
#define WAIT_TIMEOUT &H102                  'タイムアウト時間が経過したことを示す

#define INFINITE &HFFFF                     '無限に中断
#define ERROR_ALREADY_EXISTS 183            '既に存在している

#define QS_HOTKEY &H80                      'WM_HOTKEY
#define QS_KEY &H1                          'WM_KEYUP･WM_KEYDOWN･WM_SYSKEYUP･
                                            WM_SYSKEYDOWN
#define QS_MOUSEBUTTON &H4                  'WM_LBUTTONUP･WM_RBUTTONDOWN
#define QS_MOUSEMOVE &H2                    'WM_MOUSEMOVE
#define QS_PAINT &H20                       'WM_PAINT
#define QS_POSTMESSAGE &H8                  'キューに何らかのメッセージが入った
#define QS_SENDMESSAGE &H40                 '他のスレッドから送られたメッセージがある
#define QS_TIMER &H10                       'WM_TIMER
#define QS_MOUSE (&H2 Or &H4)               '(QS_MOUSEMOVE Or QS_MOUSEBUTTON)
#define QS_INPUT (&H2 Or &H4 Or &H1)        '(QS_MOUSE Or QS_KEY)
#define QS_ALLEVENTS (&H2 Or &H4 Or &H1 Or &H8 Or &H10 Or &H20 Or &H80)    '(QS_INPUT
                                            Or QS_POSTMESSAGE Or QS_TIMER Or QS_PAINT Or
                                            QS_HOTKEY)
#define QS_ALLINPUT (&H40 Or &H20 Or &H10 Or &H8 Or &H4 Or &H2 Or &H80 Or &H1)
                                            '(QS_SENDMESSAGE Or QS_PAINT Or QS_TIMER Or
                                            QS_POSTMESSAGE Or QS_MOUSEBUTTON Or
                                            QS_MOUSEMOVE Or QS_HOTKEY Or QS_KEY)
'「待機可能」タイマオブジェクトを作成
Declare Function Api_CreateWaitableTimer& Lib "kernel32" Alias "CreateWaitableTimerA"
(ByVal lpSemaphoreAttributes&, ByVal bManualReset&, ByVal lpName$)

' 既存の名前付き「待機可能」タイマオブジェクトのハンドルを取得
Declare Function Api_OpenWaitableTimer& Lib "kernel32" Alias "OpenWaitableTimerA" (ByVal
```

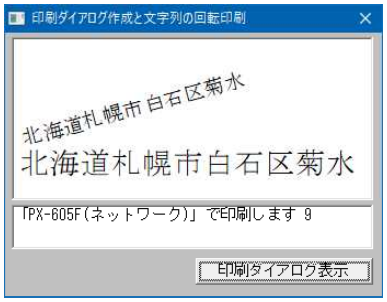
```
dwDesiredAccess&, ByVal bInheritHandle&, ByVal lpName$)

' 指定した「待機可能」タイマをアクティブにする
Declare Function Api_SetWaitableTimer& Lib "kernel32" Alias "SetWaitableTimer" (ByVal
hTimer&, lpDueTime As FILETIME, ByVal lPeriod&, ByVal pfnCompletionRoutine&, ByVal
lpArgToCompletionRoutine&, ByVal fResume&)

' 指定した「待機可能」タイマをアクティブでない状態に設定
Declare Function Api_CancelWaitableTimer& Lib "kernel32" Alias "CancelWaitableTimer"
(ByVal hTimer&)

' 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&)

' 指定したオブジェクトのいずれか1つまたはすべてがシグナル状態になったとき、またはタイムアウト時間が経過したとき制御を戻す
Declare Function Api_MsgWaitForMultipleObjects& Lib "user32" Alias
"MsgWaitForMultipleObjects" (ByVal nCount&, pHandles&, ByVal fWaitAll&, ByVal
dwMilliseconds&, ByVal dwWakeMask&)

' 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&)

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 sWait(lNumberOfSeconds As Long)
Sub sWait(lNumberOfSeconds As Long)
    Var ft As FILETIME
    Var Busy As Long
    Var Delay As Double
    Var DelayLow As Double
    Var Units As Double
    Var hTimer As Long
    Var Ret As Long

    hTimer = Api_CreateWaitableTimer(0, True, "TestTimer")
    ft.dwLowDateTime = -1
    ft.dwHighDateTime = -1

    '「待機可能」タイマをアクティブ
    Ret = Api_SetWaitableTimer(hTimer, ft, 0, 0, 0, 0)

    'ユニットをナノ秒に変換
    Units = CDbl(&H10000) * CDbl(&H10000)
    Delay = CDbl(lNumberOfSeconds) * 1000 * 10000

    ft.dwHighDateTime = -CLng(Delay / Units) - 1
    DelayLow = -Units * (Delay / Units - Fix(Delay / Units))

    If DelayLow < CDbl(-2147483648) Then
        DelayLow = Units + DelayLow
        ft.dwHighDateTime = ft.dwHighDateTime + 1
    End If

    ft.dwLowDateTime = CLng(DelayLow)

    '「待機可能」タイマをアクティブ
    Ret = Api_SetWaitableTimer(hTimer, ft, 0, 0, 0, False)

    Do
        Busy = Api_MsgWaitForMultipleObjects(1, hTimer, False, INFINITE, QS_ALLINPUT)
        CallEvent
    Loop Until Busy = WAIT_OBJECT_0
```

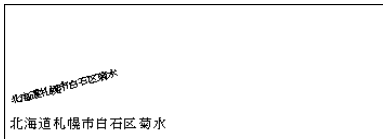
```
    Ret = Api_CloseHandle(hTimer)
End Sub

'=====================================
'= SetWaitableTimerを使用した一時停止
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    Button1.EnableWindow 0
    sWait 10
    Button1.EnableWindow -1
End Sub

'=====================================
'= Sleepを使用した一時停止
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Ret As Long

    Button2.EnableWindow 0
    Api_Sleep 10000
    Button2.EnableWindow -1
End Sub

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

印刷ダイアログ作成と文字列の回転印刷

ピクチャボックスに回転文字を描画し、印刷ダイアログボックスを表示、プリンタを選択し回転文字を印刷します。
PrintDlg 印刷ダイアログボックスを作成
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
MoveMemory メモリの指定領域をコピー
GlobalLock ヒープに確保されたメモリをロック
GlobalUnlock メモリブロックのロックを解除
GlobalFree メモリブロックのロックを解放
OpenPrinter プリンタオブジェクトをオープン
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
StartDoc 印刷ジョブを開始
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
EndDoc 印刷ジョブを終了
DeleteDC 指定されたデバイスコンテキストを削除
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
GetDeviceCaps デバイス固有の情報を取得
CreateFont 指定の属性を持つ論理フォントを作成
TextOut 文字を描画
MulDiv 2つの符号付き32ビット整数を乗算(64ビット)し、その結果を1つの符号付き32ビット整数で除算
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
FillRect ブラシで矩形領域を塗りつぶす
SetRect RECT構造体の値を設定
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
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
BitBlt ビットブロック転送を行う

「角度のある文字列を印刷」と組み合わせてみました。選択したプリンタで印刷します。　　右:印刷結果





ドキュメント名は `di.lpszDocName = StrAdr("回転文字の印刷" & Chr$(0))` で設定しています。


```
'=======================================
'= 印刷ダイアログ作成と文字列の回転印刷
'=    (PrintDlg3.bas)
'=======================================
#include "Windows.bi"

#define PD_DISABLEPRINTTOFILE &H80000      '「ファイルへ出力」チェックボックスを無効にする
#define PD_PAGENUMS &H2                    '「ページ設定」ボタンが選択された状態であることを示す
#define PD_PRINTSETUP &H40                 '「プリンタの設定」ダイアログボックスを表示する
#define PD_PRINTTOFILE &H20                '「ファイルへ出力」チェックボックスがチェック状態であることを示す

#define PD_RETURNDC &H100                  'hDCにデバイスコンテキストを格納することを表す
#define PD_RETURNDEFAULT &H400             'ダイアログを表示せず、システム設定のデフォルトプリンタで初期化

#define PD_RETURNIC &H200                  'hDCに情報コンテキストを格納することを表す
#define PD_SELECTION &H1                   '「選択した部分」ボタンが選択されていることを示す
#define PD_SHOWHELP &H800                  '「ヘルプ」ボタンを表示する
#define PD_USEDEVMODECOPIES &H40000        'PD_USEDEVMODECOPIESANDCOLLATEと同じ

Type DOCINFO
    cbSize              As Long
    lpszDocName         As Long
    lpszOutput          As Long
End Type

Type PRINTER_DEFAULTS
    pDatatype           As String * 8
    pDevMode            As Long
    DesiredAccess       As Long
End Type

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

Type PRINTDLG
    lStructSize         As Long                '構造体のサイズをバイト単位で指定
    hwndOwner           As Long                '親ウィンドウのハンドルを指定
    hDevMode            As Long                'DEVMODE構造体を含んでいるグローバルメモリオブジェクトのハンドルを指定

    hDevNames           As Long                'DEVNAMES構造体を含んだグローバルメモリオブジェクトのハンドルを指定

    hdc                 As Long                'デバイスコンテキスト
    flags               As Long                '印刷ダイアログボックスを初期化するために使われるビットフラグのセットを指定

    nFromPage           As Integer             'スタートページの初期値
    nToPage             As Integer             '最後のページの初期値
```

```
    nMinPage          As Integer            'ページ範囲の最小値
    nMaxPage          As Integer            'ページエディットコントロールの最大値
    nCopies           As Integer            '印刷部数の初期値
    hInstance         As Long               'インスタンスハンドル
    lCustData         As Long               'フック関数に渡すアプリケーション定義のデータ
    lpfnPrintHook     As Long               'フックプロシージャのポインタ
    lpfnSetupHook     As Long               'フックプロシージャのポインタ
    lpPrintTemplateName As String * 32      'ダイアログボックステンプレートの名前
    lpSetupTemplateName As String * 32      'ダイアログボックステンプレートの名前
    hPrintTemplate    As Long               'DEVNAMES構造体の文字列がデフォルトのプリンターかどうかを表す

    hSetupTemplate    As Lon                '
End Type

Type DEVNAMES
    wDriverOffset     As Integer            'デバイスドライバのファイル名を表す文字列へのオフセットアドレス
    wDeviceOffset     As Integer            'デバイスの名前を表す文字列へのオフセットアドレス
    wOutputOffset     As Integer            '出力ポートのデバイス名をあらわす文字列へのオフセットアドレス

    wDefault          As Integer            '
    extra             As String * 100       '
End Type

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               '
    dmOrientation     As Integer            'DMORIENT_PORTRAIT(縦置き)、DMORIENT_LANDSCAPE(横置き)

    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バーライド

    dmPaperWidth      As Integer            'dmPaperSizeメンバで指定した用紙の幅をオーバーライド
    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 Integer            '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

    dmICMMethod       As Long               '非ICMアプリケーションの場合に、ICMが使用可能かどうかを指定

    dmICMIntent       As Long               'カラーマッチング方法のデフォルトを指定
    dmMediaType       As Long               '印刷メディアのタイプを指定
    dmDitherType      As Long               'ディザリング方法を指定
    dmReserved1       As Long               '予約済み(0)
    dmReserved2       As Long               '予約済み(0)
    dmPanningWidth    As Long               'NT系(0)
    dmPanningHeight   As Long               'NT系(0)
End Type

' 印刷ダイアログボックスを作成
Declare Function Api_PrintDlg& Lib "comdlg32" Alias "PrintDlgA" (lpPrintdlg As PRINTDLG)
```

```
' 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&)

' 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&)

' 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&)

' メモリブロックのロックを解放
Declare Function Api_GlobalFree& Lib "kernel32" Alias "GlobalFree" (ByVal hMem&)

' プリンタオブジェクトをオープン
Declare Function Api_OpenPrinter& Lib "winspool.drv" Alias "OpenPrinterA" (ByVal
pPrinterName$, phPrinter&, pDefault As PRINTER_DEFAULTS)

' プリンタオブジェクトを閉じる
Declare Function Api_ClosePrinter& Lib "winspool.drv" Alias "ClosePrinter" (ByVal
hPrinter&)

' 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&)

' 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$, ByVal
lpDevName$, ByVal lpOutput$, ByVal lpInitData&)

' デバイス固有の情報を取得
Declare Function Api_GetDeviceCaps& Lib "gdi32" Alias "GetDeviceCaps" (ByVal hDC&, ByVal
nIndex&)

' 指定の属性を持つ論理フォントを作成
Declare Function Api_CreateFont& Lib "gdi32" Alias "CreateFontA" (ByVal Hei&, ByVal Wid&,
ByVal Esc&, ByVal Ori&, ByVal Wei&, ByVal Ita&, ByVal Und&, ByVal Stk&, ByVal Chr&, ByVal
Out&, ByVal Clp&, ByVal Qua&, ByVal Pit&, ByVal Face$)

' 文字を描画
Declare Function Api_TextOut& Lib "gdi32" Alias "TextOutA" (ByVal hDC&, ByVal nXStart&,
ByVal nYStart&, ByVal lpString$, ByVal cbString&)

' 2つの符号付き32ビット整数を乗算(64ビット)し、その結果を1つの符号付き32ビット整数で除算
Declare Function Api_MulDiv& Lib "Kernel32" Alias "MulDiv" (ByVal nNumber&, ByVal
nNumerator&, ByVal nDenominator&)

' 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&)
```

```
' 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&)

' 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&)

' 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&)

' 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 PS_SOLID 0
#define PRINTER_ACCESS_ADMINISTER &H4      'プリンタアクセス権の管理者権限を示す
#define PRINTER_ACCESS_USE &H8             'プリンタアクセス権のユーザー権限を示す
#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
#define DEFAULT_PITCH 0                    '文字ピッチデフォルト
#define PROOF_QUALITY 2                    'フォントの文字品質が、論理フォントの属性を正確に一致さ
                                           せることよりも重視
#define CLIP_DEFAULT_PRECIS 0
#define OUT_DEFAULT_PRECIS 0
#define FW_NORMAL 400
#define DEFAULT_CHARSET 1
#define TRANSPARENT 1                      '背景色を設定しない
#define LOGPIXELSX 88                      'スクリーン幅において1論理インチあたりのピクセル数
#define LOGPIXELSY 90                      'スクリーン高さにおいて1論理インチあたりのピクセル数
#define COLOR_WINDOW 5                     'ウィンドウの背景色
#define SRCCOPY &HCC0020                   'そのまま転送
#define MSG "北海道札幌市白石区菊水"

Var Shared Text1 As Object
Var Shared Picture1 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Picture1.Attach GetDlgItem("Picture1")
Button1.Attach getDlgItem("Button1") : Button1.SetFontSize 14

Var Shared hDC As Long
Var Shared mDC As Long
Var Shared mBitmap As Long
Var Shared DevName As String

'=====================================
'=
'=====================================
Declare Function CreateMyFont(nSize As Integer, nDegrees As Long) As Long
Function CreateMyFont(nSize As Integer, nDegrees As Long) As Long
    CreateMyFont = Api_CreateFont(-Api_MulDiv(nSize, Api_GetDeviceCaps(Api_GetDC(0),
LOGPIXELSY), 72), 0, nDegrees * 10, 0, FW_NORMAL, 0, 0, 0, DEFAULT_CHARSET,
OUT_DEFAULT_PRECIS, CLIP_DEFAULT_PRECIS, PROOF_QUALITY, DEFAULT_PITCH, "MS 明朝")
End Function

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

```
    mDC = Api_CreateCompatibleDC(Api_GetDC(Picture1.GethWnd))
    mBitmap = Api_CreateCompatibleBitmap(Api_GetDC(Picture1.GethWnd),
Picture1.GetWidth, Picture1.GetHeight)

    Ret = Api_SelectObject(mDC, mBitmap)
    Ret = Api_SetRect(rct, 0, 0, Picture1.GetWidth, Picture1.GetHeight)
    Ret = Api_FillRect(mDC, rct, Api_GetSysColorBrush(COLOR_WINDOW))

    Ret = Api_DeleteObject(Api_SelectObject(mDC, CreateMyFont(14, 15)))
    Ret = Api_TextOut(mDC, 5, 80, MSG, Len(MSG))

    Ret = Api_DeleteObject(Api_SelectObject(mDC, CreateMyFont(20, 0)))
    Ret = Api_TextOut(mDC, 5, 100, MSG, Len(MSG))
End Sub

'=====================================
'= 印刷
'=====================================
Declare Sub InsatuOn edecl ()
Sub InsatuOn()
    Var di As DOCINFO
    Var pd As PRINTER_DEFAULTS
    Var rct As RECT
    Var hPrinter As Long
    Var phDC As Long
    Var ex As Long
    Var Ret As Long

    'フォーム描画とプリントアウト時の倍率
    ex = 5

    'プリンタ初期化
    pd.pDatatype = Chr$(0)
    pd.pDevMode = 0
    pd.DesiredAccess = PRINTER_ACCESS_ADMINISTER Or PRINTER_ACCESS_USE

    'プリンタオープン
    Ret = Api_OpenPrinter(DevName, hPrinter, pd)

    'プリンタデバイスコンテキストを取得
    phDC = Api_CreateDC("WinSpool", DevName, vbNullString, 0)
    If phDC = 0 Then GoTo *Cleanup

    di.cbSize = Len(di)
    di.lpszDocName = StrAdr("回転文字の印刷" & Chr$(0))
    di.lpszOutput = 0

    '印刷プロセス開始
    Ret = Api_StartDoc(phDC, di)
    Ret = Api_StartPage(phDC)

    '斜め15°の回転文字印刷
    Ret = Api_DeleteObject(Api_SelectObject(phDC, CreateMyFont(14 * ex, 15)))
    Ret = Api_TextOut(phDC, 5, 800, MSG, Len(MSG))

    '水平の文字印刷
    Ret = Api_DeleteObject(Api_SelectObject(phDC, CreateMyFont(20 * ex, 0)))
    Ret = Api_TextOut(phDC, 5, 1000, MSG, Len(MSG))

    '印刷プロセス終了
    Ret = Api_EndPage(phDC)
    If Ret >= 0 Then Ret = Api_EndDoc(phDC)
    Ret = Api_ClosePrinter(hPrinter)

*Cleanup
    If phDC <> 0 Then Ret = Api_DeleteDC(phDC)
End Sub
```

```
'====================================
'=
'====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    ShowWindow -1
    hDC = Api_GetDC(Picture1.GethWnd)
    DrawText
End Sub

'====================================
'= 印刷ダイアログ作成とプリンタ選択
'====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var pMemMode As Long
    Var pMemName As Long
    Var pd As PRINTDLG
    Var dm As DEVMODE
    Var dn As DEVNAMES
    Var Ret As Long

    '印刷ダイアログを表示
    pd.hInstance = GethInst
    pd.hwndOwner = GethWnd
    pd.lStructSize = 66                    'Len(pd)

    pd.flags = PD_RETURNDC                 '印刷ダイアログ

    '適当に指定している
    pd.nFromPage = 1
    pd.nToPage = 15
    pd.nMinPage = 1
    pd.nMaxPage = 15

    dm.dmPaperSize = 13

    Ret = Api_PrintDlg(pd)

    '「印刷」<> 0、「キャンセル」= 0
    If Ret = 0 Then
        Text1.SetWindowText "「キャンセル」が選択されました"
        Ret = Api_GlobalFree(pd.hDevMode)
        Ret = Api_GlobalFree(pd.hDevNames)
        Exit Sub
    End If
    pMemMode = Api_GlobalLock(pd.hDevMode)
    MoveMemory dm, ByVal pMemMode, Len(dm)
    Ret = Api_GlobalUnlock(pd.hDevMode)
    Ret = Api_GlobalFree(pd.hDevMode)

    pMemName = Api_GlobalLock(pd.hDevNames)
    MoveMemory dn, ByVal pMemName, Len(dn)
    Ret = Api_GlobalUnlock(pd.hDevNames)
    Ret = Api_GlobalFree(pd.hDevNames)

    DevName = Left$(dm.dmDeviceName, InStr(dm.dmDeviceName, Chr$(0)) - 1)
    Text1.SetWindowText "「" & DevName & "」で印刷します" & Str$(dm.dmPaperSize)
    Ret = Api_DeleteObject(pd.hdc)

    InsatuOn
End Sub

'====================================
'= 再描画
'====================================
Declare Sub MainForm_MouseMove edecl ()
Sub MainForm_MouseMove()
    Var Ret As Long
```

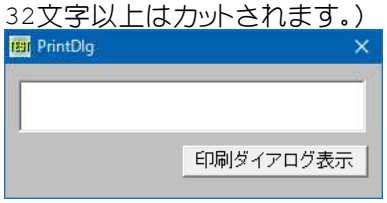
```
    Ret = Api_BitBlt(hDC, 0, 0, Picture1.GetWidth, Picture1.GetHeight, mDC, 0, 0, SRCCOPY)
End Sub

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

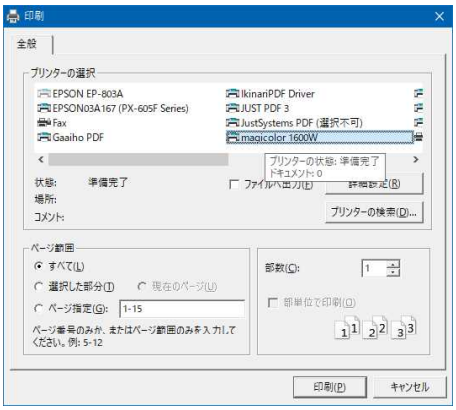
印刷ダイアログ表示とプリンタ名取得

印刷ダイアログボックスを表示、プリンタを選択し「印刷」をクリックした時そのプリンタ名を取得します。
PrintDlg 印刷ダイアログボックスを作成
DeleteObject 論理オブジェクトを削除し、全てのリソースを解放
MoveMemory メモリの指定領域をコピー
GlobalLock ヒープに確保されたメモリをロック
GlobalUnlock メモリブロックのロックを解除
GlobalFree メモリブロックのロックを解放

```
Declare Function Api_CreateDC& Lib "gdi32" Alias "CreateDCA" (ByVal lpDriverName$, ByVal lpDevName$, ByVal lpOutput$, ByVal lpInitData&)
```

等でのlpDevName$はプロパティで表示される"EPSON PX-G920"、"KONICA MINOLTA magicolor 2400W"などが必要になります。
例では、「印刷ダイアログ表示」をクリックするとデフォルトのプリンタにチェックが付いた印刷ダイアログが表示されます。
希望するプリンタを選択し、「印刷」ボタンをクリックするとlpDevName$で必要なプリンタ名が取得されます。(但し、32文字以上はカットされます。)






```
'=====================================
'= 印刷ダイアログ表示とプリンタ名取得
'=    (PrintDlg.bas)
'=====================================
#include "Windows.bi"

#define PD_COLLATE &H10                         '入力時に「部単位で印刷」チェックボックスがセットされる
#define PD_DISABLEPRINTTOFILE &H80000           '「ファイルへ出力」チェックボックスを無効にする
#define PD_ENABLEPRINTHOOK &H1000               'lpfnHookで指定されたフック関数を有効にする
#define PD_ENABLEPRINTTEMPLATE &H4000           'hInstanceがlpPrintTemplateNameメンバで指定されたダイアログテンプレートを含むリソースのインスタンスで
#define PD_ENABLEPRINTTEMPLATEHANDLE &H10000'hPrintTemplateメンバがロード済みのダイアログボッ
```

```
                                                クステンプレートを含むメモリブロ
#define PD_ENABLESETUPHOOK &H2000               'lpfnSetupHookで指定されたフック関数を有効にする
#define PD_ENABLESETUPTEMPLATE &H8000           'hInstanceおよびlpSetupTemplateNameで指定されたダイアログボックステンプレートを使用することを示す
#define PD_ENABLESETUPTEMPLATEHANDLE &H20000 'hSetupTemplateメンバがロード済みのダイアログボックステンプレートを含むメモリブロ
#define PD_HIDEPRINTTOFILE &H100000             '「ファイルへ出力」チェックボックスを隠す
#define PD_NONETWORKBUTTON &H200000             '「ネットワーク」ボタンを無効にする
#define PD_NOPAGENUMS &H8                       '「ページ設定」ボタンと関連する編集コントロールを無効にする
#define PD_NOSELECTION &H4                      '「選択した部分」ボタンを無効にする
#define PD_NOWARNING &H80                       'デフォルトプリンタがない場合に、警告を表示しない
#define PD_PAGENUMS &H2                         '「ページ設定」ボタンが選択された状態であることを示す
#define PD_PRINTSETUP &H40                      '「プリンタの設定」ダイアログボックスを表示する
#define PD_PRINTTOFILE &H20                     '「ファイルへ出力」チェックボックスがチェック状態であることを示す
#define PD_RETURNDC &H100                       'hDCにデバイスコンテキストを格納することを表す
#define PD_RETURNDEFAULT &H400                  'ダイアログを表示せず、システム設定のデフォルトプリンタで初期化
#define PD_RETURNIC &H200                       'hDCに情報コンテキストを格納することを表す
#define PD_SELECTION &H1                        '「選択した部分」ボタンが選択されていることを示す
#define PD_SHOWHELP &H800                       '「ヘルプ」ボタンを表示する
#define PD_USEDEVMODECOPIES &H40000             'PD_USEDEVMODECOPIESANDCOLLATEと同じ
#define PD_USEDEVMODECOPIESANDCOLLATE &H40000 'プリンタドライバが複数部印刷をサポートしていない場合「部数」編集コントロールを無

Type PRINTDLG
    lStructSize       As Long               '構造体のサイズをバイト単位で指定
    hwndOwner         As Long               '親ウィンドウのハンドルを指定
    hDevMode          As Long               'DEVMODE構造体を含んでいるグローバルメモリオブジェクトのハンドルを指定
    hDevNames         As Long               'DEVNAMES構造体を含んだグローバルメモリオブジェクトのハンドルを指定
    hdc               As Long               'デバイスコンテキスト
    flags             As Long               '印刷ダイアログボックスを初期化するために使われるビットフラグのセットを指定
    nFromPage         As Integer            'スタートページの初期値
    nToPage           As Integer            '最後のページの初期値
    nMinPage          As Integer            'ページ範囲の最小値
    nMaxPage          As Integer            'ページエディットコントロールの最大値
    nCopies           As Integer            '印刷部数の初期値
    hInstance         As Long               'インスタンスハンドル
    lCustData         As Long               'フック関数に渡すアプリケーション定義のデータ
    lpfnPrintHook     As Long               'フックプロシージャのポインタ
    lpfnSetupHook     As Long               'フックプロシージャのポインタ
    lpPrintTemplateName As String * 32      'ダイアログボックステンプレートの名前
    lpSetupTemplateName As String * 32      'ダイアログボックステンプレートの名前
    hPrintTemplate    As Long               'DEVNAMES構造体の文字列がデフォルトのプリンターかどうかを表す
    hSetupTemplate    As Long               '
End Type

Type DEVNAMES
    wDriverOffset     As Integer            'デバイスドライバのファイル名を表す文字列へのオフセットアドレス
    wDeviceOffset     As Integer            'デバイスの名前を表す文字列へのオフセットアドレス
    wOutputOffset     As Integer            '出力ポートのデバイス名をあらわす文字列へのオフセットアドレス
    wDefault          As Integer            '
    extra             As String * 100       '
End Type

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               '
    dmOrientation     As Integer            'DMORIENT_PORTRAIT(縦置き)、
                                            DMORIENT_LANDSCAPE(横置き)
```

```
    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バーライド
    dmPaperWidth      As Integer            'dmPaperSizeメンバで指定した用紙の幅をオーバーライド
    dmScal            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 Integer            '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
    dmICMMethod       As Long               '非ICMアプリケーションの場合に、ICMが使用可能かどうかを指定
    dmICMIntent       As Long               'カラーマッチング方法のデフォルトを指定
    dmMediaType       As Long               '印刷メディアのタイプを指定
    dmDitherType      As Long               'ディザリング方法を指定
    dmReserved1       As Long               '予約済み(0)
    dmReserved2       As Long               '予約済み(0)
    dmPanningWidth    As Long               'NT系(0)
    dmPanningHeight   As Long               'NT系(0)
End Type

' 印刷ダイアログボックスを作成
Declare Function Api_PrintDlg& Lib "comdlg32" Alias "PrintDlgA" (lpPrintdlg As PRINTDLG)

' 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&)

' 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&)

' 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&)

' メモリブロックのロックを解放
Declare Function Api_GlobalFree& Lib "kernel32" Alias "GlobalFree" (ByVal hMem&)

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 hMemMode As Long
    Var pMemMode As Long
    Var hMemName As Long
    Var pMemName As Long
    Var pd As PRINTDLG
```

```
    Var dm As DEVMODE
    Var dn As DEVNAMES
    Var Ret As Long

    '印刷ダイアログを表示
    pd.hInstance = GethInst
    pd.hwndOwner = GethWnd
    pd.lStructSize = 66                         'Len(pd)
    pd.hDevMode = hMemMode
    pd.hDevNames = hMemName
    pd.flags = PD_RETURNDC                      'PD_DISABLEPRINTTOFILE Or PD_RETURNDC Or
                                                PD_ALLPAGES Or PD_PAGENUMS Or
                                                PD_USEDEVMODECOPIES
    pd.nFromPage = 1
    pd.nToPage = 15
    pd.nMinPage = 1
    pd.nMaxPage = 15

    Ret = Api_PrintDlg(pd)

    '「印刷」<> 0、「キャンセル」= 0
    If Ret = 0 Then
        Text1.SetWindowText "「キャンセル」が選択されました"
        Ret = Api_GlobalFree(pd.hDevMode)
        Ret = Api_GlobalFree(pd.hDevNames)
        Exit Sub
    End If

    pMemMode = Api_GlobalLock(pd.hDevMode)
    MoveMemory dm, ByVal pMemMode, Len(dm)
    Ret = Api_GlobalUnlock(pd.hDevMode)
    Ret = Api_GlobalFree(pd.hDevMode)

    pMemName = Api_GlobalLock(pd.hDevNames)
    MoveMemory dn, ByVal pMemName, Len(dn)
    Ret = Api_GlobalUnlock(pd.hDevNames)
    Ret = Api_GlobalFree(pd.hDevNames)

    Text1.SetWindowText "「" & Left$(dm.dmDeviceName, InStr(dm.dmDeviceName, Chr$(0)) -
1) & "」が選択されました"
    Ret = Api_DeleteObject(pd.hdc)
End Sub

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

印刷デバイスの曲線描画能力を取得

EnumPrinters 使用可能なプリンタ・プリントサーバーなどを列挙
CopyMemory メモリブロックを別の領域に移動
GetDeviceCaps デバイス固有の情報を取得
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
CURVECAPS(28) デバイスがサポートする曲線描画能力

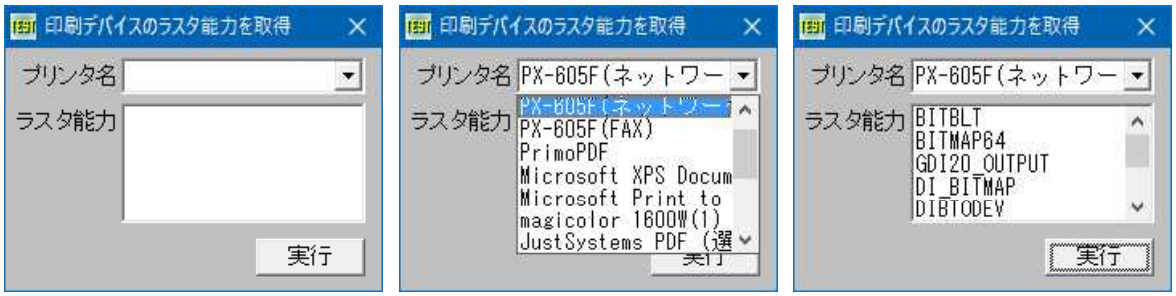

```
'=====================================
'= 印刷デバイスの曲線描画能力を取得
'=    (CURVECAPS.bas)
'=====================================
#include "Windows.bi"

Type PRINTER_INFO_5
    pPrinterName      As Long
    pPortName         As Long
    Attributes        As Long
    DeviceNotSelectedTimeOut As Long
    TransmissionRetryTimeOut As Long
End Type

#define PRINTER_ENUM_NAME &H8              'Nameで指定されたプリンタを列挙
#define MAX_DEVICENAME 64                  '
#define CURVECAPS 28                       'デバイスがサポートする曲線描画能力
#define CC_CHORD 4                         '弓形をサポート
#define CC_CIRCLES 1                       '円をサポート
#define CC_ELLIPSES 8                      '楕円をサポート
#define CC_INTERIORS 128                   '内部の塗りつぶしをサポート
#define CC_NONE 0                          '曲線をサポートしない
#define CC_PIE 2                           '扇形をサポート
#define CC_ROUNDRECT 256                   '角の丸い矩形をサポート
#define CC_STYLED 32                       'スタイルを持つ境界線をサポート
#define CC_WIDE 16                         '太い境界線をサポート
#define CC_WIDESTYLED 64                   'スタイルを持つ太い境界線をサポート

' 使用可能なプリンタ・プリントサーバーなどを列挙する
Declare Function Api_EnumPrinters& Lib "winspool.drv" Alias "EnumPrintersA" (ByVal
Flags&, ByVal Name$, ByVal Level&, pPrinterEnum As Any, ByVal cbBuf&, pcbNeeded&,
pcReturned&)

' メモリブロックを別の領域に移動する
Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (Destination As Any, Source
As Any, ByVal Length&)

' 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$, ByVal
lpDevName$, ByVal lpOutput$, ByVal lpInitData&)

Var Shared Combo1 As Object
Var Shared List1 As Object

Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
List1.SetWindowSize 206, 90

Var Shared PrinterName As String

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

```
    Var PrintServer As String
    Var Need As Long
    Var Returned As Long
    Var Level As Long
    Var CT As Long
    Var Ret As Long

    'ローカルプリンタから検索(サーバー名 ¥¥xxx 指定可)
    PrintServer = ""

    'PRINTER_INFO_5構造体を受け取る
    Level = 5

    'バッファに必要なバイト数を調べる
    Ret = Api_EnumPrinters(PRINTER_ENUM_NAME, PrintServer, Level, Chr$(0), 0, Need,
Returned)
    If Need = 0 Then End

    '全プリンタ情報を得る
    Var Buffer(Need - 1) As byte
    Ret = Api_EnumPrinters(PRINTER_ENUM_NAME, PrintServer, Level, Buffer(0), Need, Need,
Returned)

    '構造体リストの準備
    Var PI_5(Returned - 1) As PRINTER_INFO_5

    For CT = 0 To Returned - 1

        'バッファから構造体1つ分を抜き取る
        CopyMemory PI_5(CT), Buffer(CT * Len(PI_5(CT))), Len(PI_5(CT))

        'プリンタ名を得る
        PrinterName = String$(MAX_DEVICENAME, Chr$(0))
        CopyMemory PrinterName, ByVal PI_5(CT).pPrinterName, Len(PrinterName)
        PrinterName = kLeft$(PrinterName$, KInStr(1, PrinterName, Chr$(0)) - 1)

        Combo1.AddString PrinterName
    Next
End Sub

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

    'プリンタ名から、そのデバイスコンテキストを作成
    phDC = Api_CreateDC("WinSpool", PrinterName, ByVal 0, 0)

    'リストボックスを初期化
    List1.Resetcontent

    '曲線描画能力を取得
    Ret = Api_GetDeviceCaps(phDC, CURVECAPS)

    '曲線描画能力を表示
    If Ret And CC_CIRCLES    Then List1.AddString "CIRCLES"
    If Ret And CC_PIE       Then List1.AddString "PIE"
    If Ret And CC_CHORD     Then List1.AddString "CHORD"
    If Ret And CC_ELLIPSES   Then List1.AddString "ELLIPSES"
    If Ret And CC_WIDE      Then List1.AddString "WIDE"
    If Ret And CC_STYLED     Then List1.AddString "STYLED"
    If Ret And CC_WIDESTYLED Then List1.AddString "WIDESTYLED"
    If Ret And CC_INTERIORS  Then List1.AddString "INTERIORS"
    If Ret And CC_ROUNDRECT  Then List1.AddString "ROUNDRECT"
End Sub
```

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

印刷デバイスのラスタ能力を取得

EnumPrinters 使用可能なプリンタ･プリントサーバーなどを列挙
MoveMemory メモリの指定領域をコピー
GetDeviceCaps デバイス固有の情報を取得
CreateDC デバイスコンテキストを、指定された名前で作成




```
'====================================
'= 印刷デバイスのラスタ能力を取得
'=    (PrinterRastercap.bas)
'====================================
#include "Windows.bi"

Type PRINTER_INFO_5
    pDeviceName       As Long
    pPortName         As Long
    Attributes        As Long
    DeviceNotSelectedTimeOut As Long
    TransmissionColorDeviceryTimeOut As Long
End Type

' 使用可能なプリンタ･プリントサーバーなどを列挙する
Declare Function Api_EnumPrinters& Lib "winspool.drv" Alias "EnumPrintersA" (ByVal
Flags&, ByVal Name$, ByVal Level&, pPrinterEnum As Any, ByVal cbBuf&, pcbNeeded&,
pcReturned&)

' 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&)

' 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)

#define PRINTER_ENUM_DEFAULT &H1          'デフォルトのプリンタに関する情報を列挙
#define PRINTER_ENUM_LOCAL &H2            'Nameの設定を無視して、ローカルプリンタを列挙
#define PRINTER_ENUM_NAME &H8             'Nameで指定されたプリンタを列挙
#define PRINTER_ENUM_SHARED &H20          '共有属性を持つプリンタを列挙
#define MAX_DEVICENAME 64
#define DC_COLORDEVICE 32

#define RASTERCAPS 38                     'ラスタ能力(戻り値は以下の定数の組み合わせ)
#define RC_BANDING 2                      'バンド処理のサポートが必要
```

```
#define RC_BIGFONT &H400                    '大きいフォントをサポート
#define RC_BITBLT 1                         'ビットマップの転送をサポート
#define RC_BITMAP64 8                       '64KBより大きいビットマップをサポート
#define RC_DEVBITS &H8000                   '
#define RC_DI_BITMAP &H80                   'SetDIBits関数とGetDIBits関数をサポート
#define RC_DIBTODEV &H200                   'SetDIBitsToDevice関数をサポート
#define RC_FLOODFILL &H1000                 '塗りつぶしをサポート
#define RC_GDI20_OUTPUT &H10                'Windows2.0の機能をサポート
#define RC_GDI20_STATE &H20                 '
#define RC_OP_DX_OUTPUT &H4000              '
#define RC_PALETTE &H100                    'パレットベースのデバイス
#define RC_SAVEBITMAP &H40                  '
#define RC_SCALING 4                        'スケーリングをサポート
#define RC_STRETCHBLT &H800                 'StretchBlt関数をサポート
#define RC_STRETCHDIB &H2000                'StretchDIBits関数をサポート

Var Shared Text(1) As Object
Var Shared Combo1 As Object
Var Shared List1 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
Next
Combo1.Attach GetDlgItem("Combo1") : Combo1.SetFontSize 14
List1.Attach GetDlgItem("List1") : List1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var Shared DeviceName As String

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var PrintServer As String
    Var Needed As Long
    Var Returned As Long
    Var Level As Long
    Var CT As Long
    Var Device As Long

    'ローカルプリンタから検索
    PrintServer = ""

    'PRINTER_INFO_5構造体を受け取る
    Level = 5

    'バッファに必要なバイト数を調べる
    Device = Api_EnumPrinters(PRINTER_ENUM_NAME, PrintServer, Level, Chr$(0), 0, Needed,
Returned)
    If Needed = 0 Then End

    '全プリンタ情報を得る
    Var Buffer(Needed - 1) As byte
    Device = Api_EnumPrinters(PRINTER_ENUM_NAME, PrintServer, Level, Buffer(0), Needed,
Needed, Returned)

    '構造体リストの準備
    Var PI_5(Returned - 1) As PRINTER_INFO_5

    For CT = 0 To Returned - 1

        'バッファから構造体1つ分を抜き取る
        MoveMemory PI_5(CT), Buffer(CT * Len(PI_5(CT))), Len(PI_5(CT))

        'プリンタ名を得る
        DeviceName = String$(MAX_DEVICENAME, Chr$(0))
        MoveMemory DeviceName, ByVal PI_5(CT).pDeviceName, Len(DeviceName)
```

```
        DeviceName = KLeft$(DeviceName, KInStr(1, DeviceName, Chr$(0)) - 1)

        Combo1.AddString DeviceName
    Next
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var PrinterRasterCaps As Long
    Var hDC As Long

    'リストボックスを初期化
    List1.Resetcontent

    DeviceName = Combo1.GetWindowText
    If DeviceName = "" Then
        A% = MessageBox(GetWindowtext, "プリンタ名を指定してください！", 0, 2)
        Exit Sub
    End If

    hDC = Api_CreateDC(ByVal 0, DeviceName, ByVal 0, ByVal 0)

    'ラスタ能力を取得
    PrinterRasterCaps = Api_GetDeviceCaps(hDC, RASTERCAPS)

    'ラスタ能力を表示
    If PrinterRasterCaps And RC_BITBLT Then      List1.AddString "BITBLT"
    If PrinterRasterCaps And RC_BANDING Then     List1.AddString "BANDING"
    If PrinterRasterCaps And RC_SCALING Then     List1.AddString "SCALING"
    If PrinterRasterCaps And RC_BITMAP64 Then    List1.AddString "BITMAP64"
    If PrinterRasterCaps And RC_GDI20_OUTPUT Then List1.AddString "GDI20_OUTPUT"
    If PrinterRasterCaps And RC_GDI20_STATE Then  List1.AddString "GDI20_STATE"
    If PrinterRasterCaps And RC_SAVEBITMAP Then   List1.AddString "SAVEBITMAP"
    If PrinterRasterCaps And RC_DI_BITMAP Then    List1.AddString "DI_BITMAP"
    If PrinterRasterCaps And RC_PALETTE Then      List1.AddString "PALETTE"
    If PrinterRasterCaps And RC_DIBTODEV Then     List1.AddString "DIBTODEV"
    If PrinterRasterCaps And RC_BIGFONT Then      List1.AddString "BIGFONT"
    If PrinterRasterCaps And RC_STRETCHBLT Then   List1.AddString "STRETCHBLT"
    If PrinterRasterCaps And RC_FLOODFILL Then    List1.AddString "FLOODFILL"
    If PrinterRasterCaps And RC_STRETCHDIB Then   List1.AddString "STRETCHDIB"
    If PrinterRasterCaps And RC_OP_DX_OUTPUT Then List1.AddString "OP_DX_OUTPUT"
    If PrinterRasterCaps And RC_DEVBITS Then      List1.AddString "DEVBITS"
End Sub

'=====================================
'=
'=====================================
Declare Sub Combo1_Click edecl ()
Sub Combo1_Click()
    List1.Resetcontent
End Sub

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

インストール済メールアプリケーション取得

レジストリからメールアプリケーションを取得し列挙します。

RegOpenKey 指定されたレジストリキーをオープン
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

WindowXPでの例

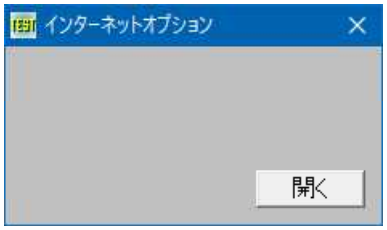
Windows2000での例

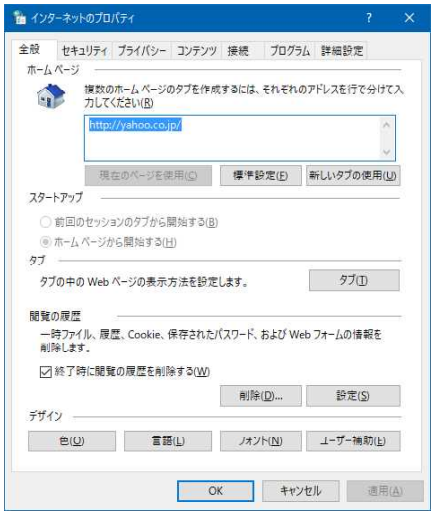
Windows10での例


```
'=====================================
'= インストール済メールアプリケーション取得
'=    (GetMailApp.bas)
'=====================================
#include "Windows.bi"

' 指定されたレジストリキーをオープン
Declare Function Api_RegOpenKey& Lib "advapi32" Alias "RegOpenKeyA" (ByVal hKey&, ByVal
lpSubKey$, phkResult&)

' 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 HKEY_LOCAL_MACHINE -2147483646     'PCを利用するユーザーに共通の設定情報
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 sKey As String * 255
    Var lRegKey As Long
    Var iKey As Integer
    Var Ret As Long
    List1.ResetContent

    Ret = Api_RegOpenKey(HKEY_LOCAL_MACHINE, "Software¥Clients¥Mail", lRegKey)

    While Api_RegEnumKey(lRegKey, iKey, sKey, 255) = 0
        List1.AddString Left$(sKey, InStr(sKey, Chr$(0)) - 1)
        iKey = iKey + 1
    Wend

    Ret = Api_RegCloseKey(lRegKey)
End Sub

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

インターネットオプション(全般)を開く

IEのインターネットオプションの全般を開きます。
ShellExecute 拡張子に関連付けされたプログラムを実行






```
'=======================================
'= インターネットオプション(全般)を開く
'=    (ShellExecute2.bas)
'=======================================
#include "Windows.bi"

' 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&)

#define vbNormalFocus 1                    '開いたアプリケーションはフォーカスを持ち、前回起動した
                                           サイズと位置に復元される
Var Shared Button1 As Object
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=======================================
'=
'=======================================
Declare Function ShowInetOption(ByVal hWnd As Long) As Integer
Function ShowInetOption(ByVal hWnd As Long) As Integer
    Var tmpStr As String
    Var Ret As Long

    tmpStr = "shell32.dll,Control_RunDLL inetcpl.cpl"

    Ret = Api_ShellExecute(hWnd, "open", "rundll32.exe", tmpStr, ByVal 0, vbNormalFocus)

    ShowInetOption = (Ret > 32)
End Function

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
   If Not ShowInetOption(GethWnd) Then
      A% = MessageBox("Error", "インターネットオプションは見つかりません！", 0, 2)
   End If
End Sub

'=======================================
'=
'=======================================
While 1
```

```
    WaitEvent
Wend
Stop
End
```

インターネット接続状況の取得(Ⅰ)

InternetGetConnectedState ローカルシステムのインターネット接続状況を取得します。

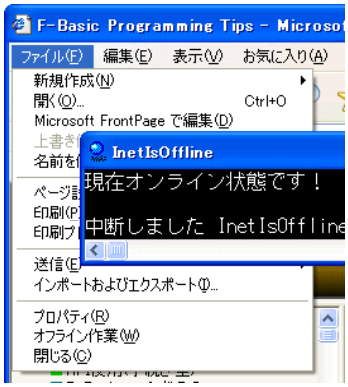

```
'=======================================
'= インターネット接続状況の取得(Ⅰ)
'=    (InternetGetConnectedState.bas)
'=======================================
#include "Windows.bi"

' ローカルシステムのインターネット接続状況を返す
Declare Function Api_InternetGetConnectedState& Lib "wininet" Alias
"InternetGetConnectedState" (ByRef lpdwFlags&, ByVal dwReserved&)

#define INTERNET_CONNECTION_OFFLINE &H20  'オフライン
#define INTERNET_CONNECTION_CONFIGURED &H40       '接続の設定を完了
#define INTERNET_CONNECTION_LAN &H2       '接続にLANを使用
#define INTERNET_CONNECTION_MODEM &H1     '接続にモデムを使用
#define INTERNET_CONNECTION_MODEM_BUSY &H8'ビジー
#define INTERNET_CONNECTION_PROXY &H4     '接続にプロキシ･サーバーを使用
#define INTERNET_RAS_INSTALLED &H10       'RAS(Remote Access Service)がインストールされている

Var Shared Text(11) As Object
Var Shared Button1 As Object

For i = 0 To 11
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1)))
    Text(i).SetFontSize 12
Next
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 12

'=======================================
'=
'=======================================
Declare Function NetConnectByLAN() As Integer
Function NetConnectByLAN() As Integer
    Var dwflags As Long
    Var Ret As Long
    Ret = Api_InternetGetConnectedState(dwflags, 0)

    NetConnectByLAN = dwflags And INTERNET_CONNECTION_LAN
End Function

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

```
Declare Function NetConnectByModem() As Integer
Function NetConnectByModem() As Integer
    Var dwflags As Long
    Var Ret As Long

    Ret = Api_InternetGetConnectedState(dwflags, 0)

    NetConnectByModem = dwflags And INTERNET_CONNECTION_MODEM
End Function

'=====================================
'=
'=====================================
Declare Function NetConnectByProxy() As Integer
Function NetConnectByProxy() As Integer
    Var dwflags As Long
    Var Ret As Long

    Ret = Api_InternetGetConnectedState(dwflags, 0)

    NetConnectByProxy = dwflags And INTERNET_CONNECTION_PROXY
End Function

'=====================================
'=
'=====================================
Declare Function NetConnectOnline() As Integer
Function NetConnectOnline() As Integer
    NetConnectOnline = Api_InternetGetConnectedState(0, 0)
End Function

'=====================================
'=
'=====================================
Declare Function NetRASInstalled() As Integer
Function NetRASInstalled() As Integer
    Var dwflags As Long
    Var Ret As Long

    Ret = Api_InternetGetConnectedState(dwflags, 0)

    NetRASInstalled = dwflags And INTERNET_RAS_INSTALLED
End Function

'=====================================
'=
'=====================================
Declare Function GetNetConnectString() As String
Function GetNetConnectString() As String
    Var dwflags As Long
    Var MSG As String
    Var CrLf As String

    CrLf = Chr$(13, 10)

    If Api_InternetGetConnectedState(dwflags, 0) Then
        If dwflags And INTERNET_CONNECTION_CONFIGURED Then
            MSG = MSG & "ネットワークに接続。" & CrLf
        End If

        If dwflags And INTERNET_CONNECTION_LAN Then
            MSG = MSG & "ローカルシステムはLANを経由してインターネットと接続。"
        End If

        If dwflags And INTERNET_CONNECTION_PROXY Then
            MSG = MSG & "プロクシーサーバーを使用。"
        Else
            MSG = MSG
        End If
```

```
        If dwflags And INTERNET_CONNECTION_MODEM Then
            MSG = MSG & "ローカルシステムはMODEMを経由してインターネットと接続。"
        End If

        If dwflags And INTERNET_CONNECTION_OFFLINE Then
            MSG = MSG & "オフライン接続。"
        End If

        If dwflags And INTERNET_CONNECTION_MODEM_BUSY Then
            MSG = MSG & "モデムはビジー状態です。"
        End If

        If dwflags And INTERNET_RAS_INSTALLED Then
            MSG = MSG & "RAS(リモートアクセスサービス)はこのシステムにインストールされています。"
        End If
    Else
        MSG = "現在インターネットに接続していません。"
    End If

    GetNetConnectString = MSG
End Function

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    If NetConnectByLAN()    = 0 Then Text(6).SetWindowText "False" Else
Text(6).SetWindowText "True"
    If NetConnectByModem() = 0 Then Text(7).SetWindowText "False" Else
Text(7).SetWindowText "True"
    If NetConnectByProxy() = 0 Then Text(8).SetWindowText "False" Else
Text(8).SetWindowText "True"
    If NetConnectOnline()   = 0 Then Text(9).SetWindowText "False" Else
Text(9).SetWindowText "True"
    If NetRASInstalled()    = 0 Then Text(10).SetWindowText "False" Else
Text(10).SetWindowText "True"
    Text(11).SetWindowText GetNetConnectString()
End Sub

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

インターネット接続状況の取得(II)

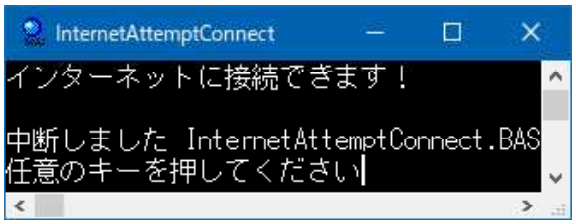
InetIsOffline システムがインターネットに接続しているかどうかを判断



```
'=====================================
'= インターネット接続状況の取得(II)
'=    (InetIsOffline.bas)
'=====================================

' システムがインターネットに接続しているかどうかを判断
Declare Function Api_InetIsOffline& Lib "url" Alias "InetIsOffline" (ByVal dwFlags&)

If Api_InetIsOffline(0) = 0 Then
    Print "現在オンライン状態です！"
Else
    Print "現在オフライン状態です！"
End If

Stop
End
```

インターネットに接続できるかどうか調べる

InternetAttemptConnect インターネットに接続できるかどうか調べる

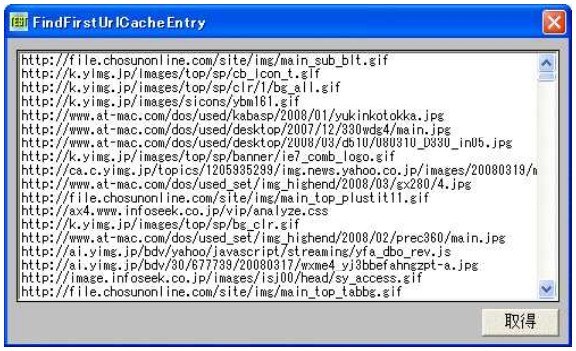


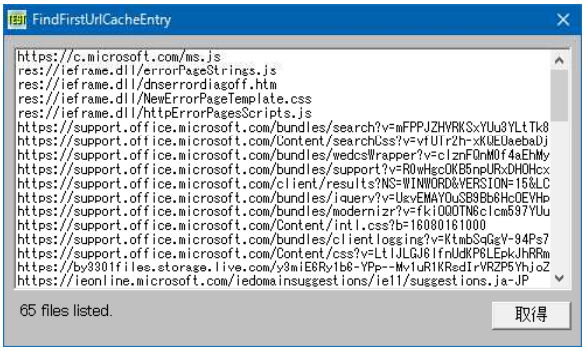

```
'=====================================
'= インターネットに接続できるかどうか調べる
'=    (InternetAttemptConnect.bas)
'=====================================
#include "Windows.bi"

' インターネットに接続できるかどうか調べる
Declare Function Api_InternetAttemptConnect& Lib "wininet" Alias
"InternetAttemptConnect" (ByVal dwReserved&)
If Api_InternetAttemptConnect(0) = 0 Then
    Print "インターネットに接続できます！"
Else
    Print "インターネットに接続できません！"
End If

Stop
End
```

インターネットのキャッシュを取得

FindFirstUrlCacheEntryインターネットキャッシュの列挙を開始
FindNextUrlCacheEntry インターネットキャッシュの次の項目を検索
FindCloseUrlCache インターネットキャッシュの列挙ハンドルを閉じる
MoveMemory メモリの指定領域をコピー
lstrlen 指定された文字列のバイトまたは文字の長さを返す
lstrcpy 文字列をコピーする

ListBoxに入る項目数が多い場合、表示しながら取得すると時間がかかるので、全て取得するまでダミー(List2)のListBoxを表示させておき全て取得した時点で
List1に切替表示させています。
例では、約6,800項目を表示するのに通常 2秒60、全項目取得後に切り替えると 1秒60 でした。






```
'=======================================
'= ブラウザのキャッシュを取得
'=    (FindFirstUrlCacheEntry.bas)
'=======================================
#include "Windows.bi"

Type FILETIME
    dwLowDateTime     As Long
    dwHighDateTime    As Long
End Type

Type INTERNET_CACHE_ENTRY_INFO
    dwStructSize      As Long
    lpszSourceUrlName       As Long
    lpszLocalFileName       As Long
    CacheEntryType    As Long
    dwUseCount        As Long
    dwHitRate         As Long
    dwSizeLow         As Long
    dwSizeHigh        As Long
    LastModifiedTime  As FILETIME
    ExpireTime        As FILETIME
    LastAccessTime    As FILETIME
    LastSyncTime      As FILETIME
    lpHeaderInfo      As Long
    dwHeaderInfoSize  As Long
    lpszFileExtension       As Long
    dwExemptDelta     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

' インターネットキャッシュの列挙を開始
Declare Function Api_FindFirstUrlCacheEntry& Lib "wininet" Alias
"FindFirstUrlCacheEntryA" (ByVal lpszUrlSeaRethPattern$, ByRef lpFirstCacheEntryInfo As
Any, ByRef lpdwFirstCacheEntryInfoBufferSize&)

' インターネットキャッシュの次の項目を検索
Declare Function Api_FindNextUrlCacheEntry& Lib "wininet" Alias
"FindNextUrlCacheEntryA" (ByVal hEnumHandle&, ByRef lpNextCacheEntryInfo As Any, ByRef
lpdwNextCacheEntryInfoBufferSize&)

' インターネットキャッシュの列挙ハンドルを閉じる
Declare Function Api_FindCloseUrlCache& Lib "wininet" Alias "FindCloseUrlCache" (ByVal
hEnumHandle&)

' 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&)

' 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&)

' 文字列をコピーする
Declare Function Api_lstrcpy& Lib "Kernel32" Alias "lstrcpyA" (ByVal lpszString1$, ByVal
lpszString2&)

#define NORMAL_CACHE_ENTRY &H1                  '通常のキャッシュエントリー
#define URLHISTORY_CACHE_ENTRY &H200000         'URLキャッシュエントリー
#define ERROR_NO_MORE_FILES 18                  'これ以上ファイルがない
#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

Var Shared List1 As Object
Var Shared List2 As Object
Var Shared Text1 As Object
Var Shared Button1 As Object

List1.Attach GetDlgItem("List1") : List1.SetFontSize 12 : List1.SetWindowSize 478, 210
List2.Attach GetDlgItem("List2") : List2.SetFontSize 12 : List2.SetWindowSize 478, 210
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    ShowWindow -1
    List1.ShowWindow -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var EnumHandle As Long
    Var icei As INTERNET_CACHE_ENTRY_INFO
    Var BuffSize As Long
    Var Ret As Long

    BuffSize = 0
    Ret = Api_FindFirstUrlCacheEntry("", ByVal 0, BuffSize)
    Dim bICEI(BuffSize) As Byte

    icei.dwStructSize = Len(icei)

    MoveMemory bICEI(0), icei, Len(icei)

    EnumHandle = Api_FindFirstUrlCacheEntry("", bICEI(0), BuffSize)

    If EnumHandle = 0 Then
        A% = MessageBox("", "キャッシュを取得できません！", 0, 2)
        Exit Sub
    End If

    MoveMemory icei, bICEI(0), Len(icei)

    Var SourceUrlName As String
    Var LocalFileName As String

    '表示項目が多い場合時間がかかるので全て取得してから表示させている
    List1.Resetcontent
    List2.Resetcontent
    List1.ShowWindow 0
    List2.ShowWindow -1

    Do
```

```
        If icei.CacheEntryType And NORMAL_CACHE_ENTRY Then
            SourceUrlName = String$(Api_lstrlen(icei.lpszSourceUrlName), 0)
            LocalFileName = String$(Api_lstrlen(icei.lpszLocalFileName), 0)
            Ret = Api_lstrcpy(SourceUrlName, icei.lpszSourceUrlName)
            Ret = Api_lstrcpy(LocalFileName, icei.lpszLocalFileName)
            List1.AddString SourceUrlName
        End If

        BuffSize = 0

        Ret = Api_FindNextUrlCacheEntry(EnumHandle, ByVal 0, BuffSize)

        If Ret <> 0 Then Exit Do
        If BuffSize = 0 Then Exit Do

        Erase bICEI

        Var bICEI(BuffSize) As Byte

        icei.dwStructSize = Len(icei)
        MoveMemory bICEI(0), icei, Len(icei)

        Ret = Api_FindNextUrlCacheEntry(EnumHandle, bICEI(0), BuffSize)

        If Ret = 0 Then Exit Do
        If Ret = ERROR_NO_MORE_FILES Then Exit Do
        If Ret = ERROR_INSUFFICIENT_BUFFER Then Exit Do

        MoveMemory icei, bICEI(0), Len(icei)
    Loop

    List2.ShowWindow 0
    List1.ShowWindow -1

    Ret = Api_FindCloseUrlCache(EnumHandle)

    Text1.SetWindowText Str$(List1.GetCount) & " files listed."
End Sub

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

ウィンドウがUnicodeを使用しているか判断

IsWindowUnicode ウィンドウがUnicodeを使用しているかどうかを判断します。

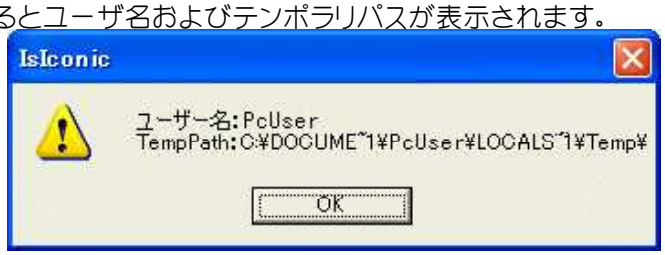

```
'========================================
'= ウィンドウがUnicodeを使用しているか判断
'=    (IsWindowUnicode.bas)
'========================================
#include "Windows.bi"
```

```
' ウィンドウがUnicodeを使用しているか判断
Declare Function Api_IsWindowUnicode& Lib "user32" Alias "IsWindowUnicode" (ByVal hWnd&)

Var Shared Edit1 As Object
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14

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

    Ret = Api_IsWindowUnicode(Edit1.GethWnd)
    If Ret <> 0 Then
        Edit1.SetWindowText "Unicodeウィンドウです"
    Else
        Edit1.SetWindowText "Unicodeウィンドウではありません"
    End If
End Sub

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

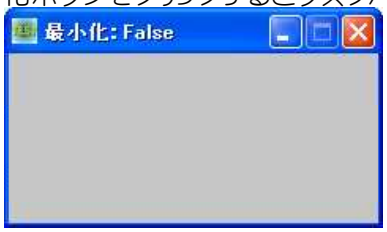
ウィンドウが最小化されているかどうかを判断

ウインドウが最小化されているかどうかの判断と、テンポラリパスを取得します。
GetUserName ユーザー名を取得
GetTempPath 一時フォルダ（テンポラリフォルダ）を取得
IsIconic ウィンドウが最小化されているかどうかを判断

タイマーを貼り付けます。起動するとユーザ名およびテンポラリパスが表示されます。



【OK】をクリックすると、フォームが最小化されていないのでメニューバーに【最小化:False】と表示されます。アイコン化ボタンをクリックするとタスクバー上で【最小化:True】と表示されます。



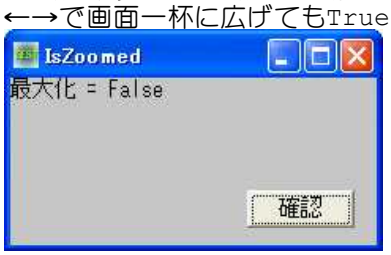

```
'=======================================
'= ウィンドウが最小化されているかどうかを判断
'=    (IsIconic.bas)
'=======================================
#include "Windows.bi"

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

```
' 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$)

' ウィンドウが最小化されているかどうかを判断する関数
Declare Function Api_IsIconic& Lib "user32" Alias "IsIconic" (ByVal hWnd&)

Var Shared Timer1 As Object
Timer1.Attach GetDlgItem("Timer1")

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var Temp As String
    Var UserName As String
    Var Ret As Long

    'バッファ初期化
    Temp = String$(100, Chr$(0))

    'テンポラリパス取得
    Ret = Api_GetTempPath(100, Temp)

    'Null除去
    Temp = Left$(Temp, InStr(Temp, Chr$(0)) - 1)

    'バッファ初期化
    UserName = String$(100, Chr$(0))

    'ユーザ名取得
    Ret = Api_GetUserName(UserName, 100)

    'Null除去
    UserName = Left$(UserName, InStr(UserName, Chr$(0)) - 1)

    'TempPathとユーザー名表示
    A% = MessageBox(GetWindowText, "ユーザー名:" & UserName & Chr$(13) & "TempPath:" & Temp,
0, 2)

    Timer1.SetInterval 10
    Timer1.Enable -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var State As String
    Var Ret As Long

    '最小化されているかをチェック
    Ret = Api_IsIconic(GethWnd)

    If Ret = 1 Then State = "True" Else State = "False"

    'フォームキャプション更新
    SetWindowText "最小化:" & State
End Sub

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

ウィンドウが最大化しているかどうかの判断

ウィンドウが最大化されているかどうかを判断します。
IsZoomed ウィンドウが最大化されているかどうかを判断する関数

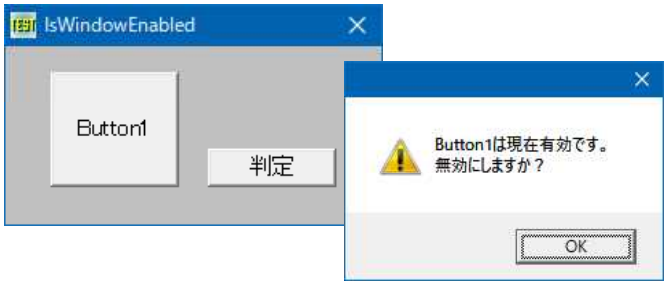
最大化ボタンで最大化した場合は「最大化 = True」になります。
←→で画面一杯に広げてもTrueにはなりません。

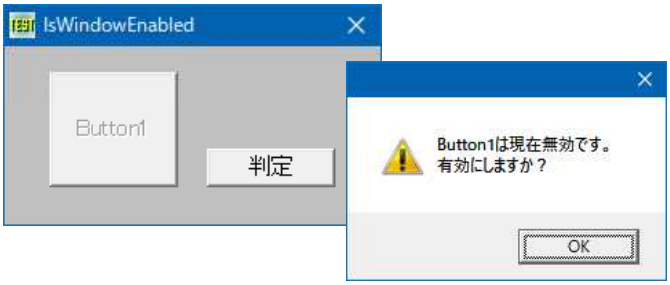



```
'=======================================
'= ウインドウが最大化しているかどうかの判断
'=    (IsZoomed.bas)
'=======================================
#include "Windows.bi"

' ウィンドウが最大化されているかどうかを判断する関数
Declare Function Api_IsZoomed& Lib "user32" Alias "IsZoomed" (ByVal hWnd&)

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

    Ret = Api_IsZoomed(GethWnd)
    If Ret Then
        Print "最大化 = True"
    Else
        Print "最大化 = False"
    End If
End Sub

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

ウィンドウが有効か無効か判別

IsWindowEnabled ウィンドウが有効か無効か判別
EnableWindow 指定されたウィンドウまたはコントロールへのマウス入力およびキーボード入力を有効または無効にする



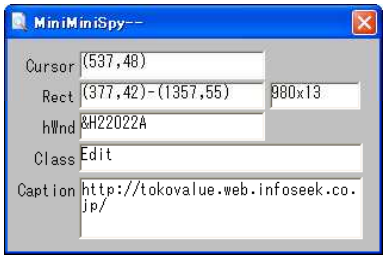


```
'=====================================
'= ウィンドウが有効か無効か判別
'=    (IsWindowEnabled.bas)
'=====================================
#include "Windows.bi"

' ウィンドウが有効か無効か判別
Declare Function Api_IsWindowEnabled& Lib "user32" Alias "IsWindowEnabled" (ByVal hWnd&)

' 指定されたウィンドウまたはコントロールへのマウス入力およびキーボード入力を有効または無効にする
Declare Function Api_EnableWindow& Lib "user32" Alias "EnableWindow" (ByVal hWnd&,
bEnable&)

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 Button2_on edecl ()
Sub Button2_on()
    Var bl As Long
    Var Ret As Long

    bl = Api_IsWindowEnabled(Button1.GethWnd)

    If bl = 0 Then
        A% = MessageBox("", "Button1は現在無効です。" & Chr$(13, 10) & "有効にしますか？", 0, 2)
        Ret = Api_EnableWindow(Button1.GethWnd, 1)
    Else
        A% = MessageBox("", "Button1は現在有効です。" & Chr$(13, 10) & "無効にしますか？", 0, 2)
        Ret = Api_EnableWindow(Button1.GethWnd, ByVal 0)
    End If
End Sub

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

ウインドウクラス情報取得（Ⅰ）

ウインドウクラスの情報を取得します。MiniMiniSpy-- はウインドウクラス情報取得（Ⅱ）をご覧ください
GetWindowRect ウィンドウの座標をスクリーン座標系で取得
GetCursorPos カーソルの現在のスクリーン座標の取得
WindowFromPoint 指定の座標位置にあるウィンドウハンドルを取得
GetClassName ウィンドウのクラス名を取得
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（親を持たないウィンドウ）のハンドルを返す
GetWindowText ウインドウのタイトル文字列を取得
SendMessage ウィンドウにメッセージを送信

カーソル位置のクラス名を取得し、そのハンドル・矩形領域・そのサイズを表示します。
選択された矩形領域を薄い枠で囲むようにしてみました。少々動作が鈍いようです…
Spy++や、このてのフリーソフトなどはどのようにしているのか解りませんが、Form2を用意し矩形領域に重ね透過処理をしています。
Form2は、タイトルバー（なし）に設定していますので青フレームが無くなり見にくいかもしれません。
サイズを合わせたPictureBoxを乗せて枠を強調してみましたが、さらに動作が鈍くなったのでやめました。
ウインドウクラス情報取得（Ⅲ）コード部下段参照
図は、右図のアドレス欄にカーソルを合わせた時の情報です。



FrontPageの編集領域を指示した場合の例


```
'=======================================
'= ウインドウクラス情報取得
'=    (MiniMiniSpy--.bas)
'=======================================
#include "Windows.bi"

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)

' カーソルの現在のスクリーン座標の取得
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&)

' ウィンドウのクラス名を取得する関数の宣言
Declare Function Api_GetClassName& Lib "user32" Alias "GetClassNameA" (ByVal hWnd&,
ByVal lpClassName$, ByVal nMaxCount&)

' 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
Declare Function Api_FindWindow& Lib "user32" Alias "FindWindowA" (ByVal lpClassName$,
ByVal lpWindowName$)

' 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&)

' 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 WM_GETTEXT &HD                          'コントロールのキャプション・テキストをバッファにコピー
```

```
Var Shared Timer1 As Object
Var Shared Text(9) As Object
Var Shared Edit1 As Object

Timer1.Attach GetDlgItem("Timer1")
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

'=====================================
'= 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()
    Timer1.SetInterval 10
    Timer1.Enable -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var pa As POINTAPI
    Var rc As RECT
    Var hWnd As Long
    Var Buff As String * 128
    Var Caption As String
    Var Ret As Long

    ' カーソル位置のスクリーン座標を取得
    Ret = Api_GetCursorPos(pa)

    ' 座標を含むウィンドウのハンドルを取得
    hWnd = Api_WindowFromPoint(pa.x, pa.y)

    ' 矩形領域を取得
    Ret = Api_GetWindowRect(hWnd, rc)

    ' ウィンドウのハンドルを取得できたとき
    If hWnd <> 0 Then

        ' カーソル座標を表示
        Text(4).SetWindowText "(" & Trim$(Str$(pa.x)) & "," & Trim$(Str$(pa.y)) & ")"

        ' 矩形座標を表示
        Text(5).SetWindowText "(" & Trim$(Str$(rc.Left)) & "," & Trim$(Str$(rc.Top)) &
")-(" & Trim$(Str$(rc.Right)) & "," & Trim$(Str$(rc.Bottom)) & ")"

        ' 矩形サイズを表示
        Text(6).SetWindowText Trim$(Str$(rc.Right - rc.Left)) & "x" &
Trim$(Str$(rc.Bottom - rc.Top))
```

```
        ' ウィンドウのハンドルを表示
        Text(7).SetWindowText "&&H" & hex$(hWnd)

        ' ウィンドウのクラス名を取得
        Ret = Api_GetClassName(hWnd, Buff, Len(Buff))

        Text(8).SetWindowText TrimNull(Buff)

        ' キャプションの文字列取得表示
        Caption = space$(250)
        Ret = Api_GetWindowText(hWnd, Caption, Len(Caption))
        Ret = Api_SendMessage(hWnd, WM_GETTEXT, Len(Caption), Caption)
        Edit1.SetWindowText TrimNull(Caption)
    End If
End Sub

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

```
'=====================================
'=  ウインドウクラス情報取得(MiniMiniSpy--)
'=====================================
#include "Windows.bi"

Type POINTAPI
    x   As Long
    y   As Long
End Type


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

' 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)

' カーソルの現在のスクリーン座標の取得
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&)

' ウィンドウのクラス名を取得する関数の宣言
Declare Function Api_GetClassName& Lib "user32" Alias "GetClassNameA" (ByVal hWnd&,
ByVal lpClassName$, ByVal nMaxCount&)

' 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&)
```

```
' 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&)

' 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 WM_GETTEXT &HD                          'コントロールのキャプション・テキストをバッファにコピー
#define GWL_EXSTYLE -20                         '拡張ウィンドウスタイル
#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 MainForm As Object
Var Shared Form2 As Object
Var Shared Timer1 As Object
Var Shared Text(9) As Object
Var Shared Edit1 As Object

MainForm.Attach GethWnd
Timer1.Attach GetDlgItem("Timer1")

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

Var Shared hWnd2 As Long

'=====================================
'= 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()

    ' Form2を作成しハンドル取得
    Form2.CreateWindow "Form2", 0
    hWnd2 = Form2.GethWnd

    Timer1.SetInterval 10
    Timer1.Enable -1
End Sub
```

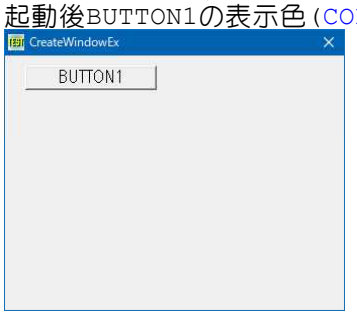
```
'====================================
'=
'====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var pa As POINTAPI
    Var rc As RECT
    Var hWnd As Long
    Var Buff As String * 128
    Var Caption As String
    Var dwStyle As Long
    Var Ret As Long

    ' カーソル位置のスクリーン座標を取得
    Ret = Api_GetCursorPos(pa)

    ' 座標を含むウィンドウのハンドルを取得
    hWnd = Api_WindowFromPoint(pa.x, pa.y)

    ' 矩形領域を取得
    Ret = Api_GetWindowRect(hWnd, rc)

    ' 取得した矩形領域にForm2のサイズを合わせる
    Form2.MoveWindow rc.Left, rc.Top
    Form2.SetWindowSize rc.Right - rc.Left, rc.Bottom - rc.Top

    ' 拡張ウィンドウスタイルにWS_EX_LAYEREDを追加する
    dwStyle = Api_GetWindowLong(hWnd2, GWL_EXSTYLE)
    dwStyle = dwStyle Or WS_EX_LAYERED

    ' Form2ウィンドウ全体を透明にする(薄い枠のみ判断できる程度)
    Ret = Api_SetWindowLong(hWnd2, GWL_EXSTYLE, dwStyle)
    Ret = Api_SetLayeredWindowAttributes(hWnd2, Form2.GetBackColor, 0, LWA_COLORKEY)
    Form2.ShowWindow -1

    ' ウィンドウのハンドルを取得できたとき
    If hWnd <> 0 Then

        ' カーソル座標を表示
        Text(4).SetWindowText "(" & Trim$(Str$(pa.x)) & "," & Trim$(Str$(pa.y)) & ")"

        ' 矩形座標を表示
        Text(5).SetWindowText "(" & Trim$(Str$(rc.Left)) & "," & Trim$(Str$(rc.Top)) &
")-(" & Trim$(Str$(rc.Right)) & "," & Trim$(Str$(rc.Bottom)) & ")"

        ' 矩形サイズを表示
        Text(6).SetWindowText Trim$(Str$(rc.Right - rc.Left)) & "x" &
Trim$(Str$(rc.Bottom - rc.Top))

        ' ウィンドウのハンドルを表示
        Text(7).SetWindowText "&&H" & hex$(hWnd)

        ' ウィンドウのクラス名を取得
        Ret = Api_GetClassName(hWnd, Buff, Len(Buff))
        Text(8).SetWindowText TrimNull(Buff)

        ' キャプション取得
        Caption = space$(250)
        Ret = Api_GetWindowText(hWnd, Caption, Len(Caption))
        Ret = Api_SendMessage(hWnd, WM_GETTEXT, Len(Caption), Caption)
        Edit1.SetWindowText TrimNull(Caption)
    End If
End Sub

'====================================
'=
'====================================
While 1
    WaitEvent
Wend
```

```
Stop
End
```

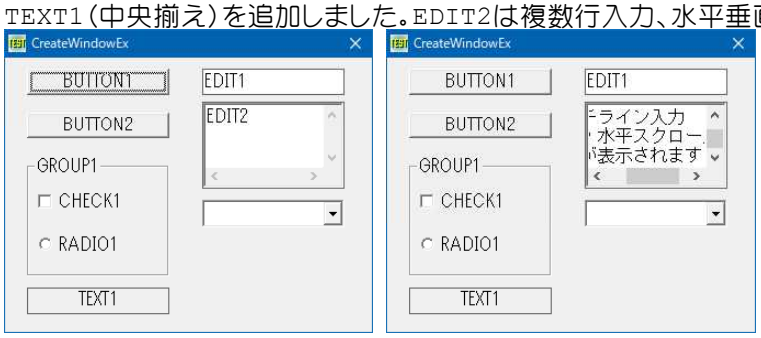
ウィンドウ(コントロール)をコードで作成

BUTTON、CHECK、GROUP、TEXT、EDIT、COMBOの各コントロールをコード上で作成します。
GetSysColor システムのCOLOR取得
CreateWindowEX 新しいウインドウ(コントロール)を作成
DestroyWindow ウインドウ(コントロール)の解放

MAINFORMは、可視なし、BUTTON1を貼り付けます。
起動後BUTTON1の表示色(COLOR_BTNFACE)を取得し、MAINFORMをその色で塗りつぶしています。



BUTTON1をクリックすることによりその他のコントロールを作成します。
TEXT1(中央揃え)を追加しました。EDIT2は複数行入力、水平垂直スクロールバー表示に設定しています。



作成するウインドウ(コントロール)	クラス名
ボタンコントロール(ボタン・ラジオ・チェック・グループ)	"Button"
エディットボックス	"Edit"
テキストボックス	"Static"
コンボボックス	"ComboBox"
リストボックス	"ListBox"
スクロールバー	"ScrollBar"

参照
コントロールをVisual Styleに<コントロールをコード(WindowXPスタイル)で作成>

```
'=====================================
'= CreateWindowExのテスト
'=    (CreateWindowEx.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&)

#define COLOR_BTNFACE 15                    '3Dオブジェクトの表面色

' 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&)

' CreateWindowExの解放
Declare Function Api_DestroyWindow& Lib "user32" Alias "DestroyWindow" (ByVal hWnd&)

#define WS_BORDER &H800000                  'フォームの枠線がある
#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_CLIPSIBLINGS &H4000000           '兄弟関係にある子ウィンドウをクリップする
#define WS_EX_WINDOWEDGE &H100              'ウィンドウが盛り上がった縁の境界線を持つように指定
#define WS_EX_OVERLAPPEDWINDOW &H300        'WS_EX_WINDOWEDGEとWS_EX_CLIENTEDGEの組み合せ
#define WS_VISIBLE &H10000000               '可視状態の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 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_EX_CLIENTEDGE &H200              '縁が沈んで見える境界線を持つウィンドウを指定
#define WS_THICKFRAME &H40000               'サイズ変更境界を持つウィンドウを作成する

#define BS_PUSHBUTTON &H0                   'プッシュボタン
#define BS_AUTOCHECKBOX &H3                 'チェックボックス
#define BS_AUTORADIOBUTTON &H9              'ラジオボタン
#define BS_GROUPBOX &H7                     'グループボックス
#define ES_AUTOHSCROLL &H80                 '入力範囲を越えたら自動的に水平スクロールする
#define ES_AUTOVSCROLL &H40                 '最後の行で自動的に垂直スクロールをする
#define ES_CENTER 1                         'テキストを水平方向で中央に表示する
#define ES_LEFT 0                           'テキストを左揃えする(デフォルト)
#define ES_LOWERCASE &H10                   '入力文字を小文字にする
#define ES_MULTILINE 4                      '複数行エディットを作成する
#define ES_NOHIDESEL &H100                  'フォーカスを失っても選択範囲を表示する
#define ES_NUMBER &H2000                    '数値入力専用にする
#define ES_OEMCONVERT &H400                 '入力文字をOEM文字セットに変換する
#define ES_PASSWORD &H20                    '入力文字をデフォルトでは*にして表示する
#define ES_READONLY &H800                   '読みとり専用にする(選択してコピーはできる)
#define ES_RIGHT 2                          'テキストを右揃えする
#define ES_UPPERCASE 8                      '入力文字を大文字にする
#define ES_WANTRETURN &H1000                '複数行エディットでEnterキーで改行する
#define CBS_SIMPLE &H1                      '単純なコンボボックス。リストは常にドロップダウン
#define CBS_DROPDOWN &H2                    'CBS_SIMPLEで、リストはドロップダウンアイコンで表示

Var Shared Button1 As Object

Button1.Attach GetDlgItem("Button1")

'=====================================
'=
'=====================================
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 StdStyles As Long
    Var Ret As Long

    'コマンドボタン作成
    Ret = Api_CreateWindowEx(0, "Button", "BUTTON2", WS_CHILD Or WS_VISIBLE Or
BS_PUSHBUTTON, 20, 50, 130, 24, GethWnd, 0, 0, 0)

    'グループ作成
    Ret = Api_CreateWindowEx(0, "Button", "GROUP1", WS_CHILD Or WS_VISIBLE Or
BS_GROUPBOX, 20, 90, 130, 110, GethWnd, 0, 0, 0)
```

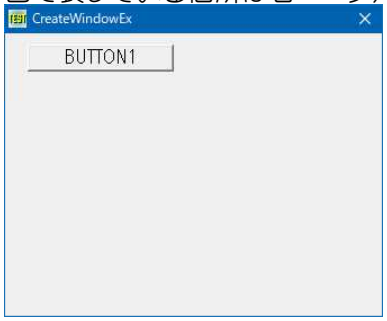
```
    'チェックボタン作成
    Ret = Api_CreateWindowEx(0, "Button", "CHECK1", WS_CHILD Or WS_VISIBLE Or
BS_AUTOCHECKBOX, 30, 120, 110, 24, GethWnd, 0, 0, 0)

    'ラジオボタン作成
    Ret = Api_CreateWindowEx(0, "Button", "RADIO1", WS_CHILD Or WS_VISIBLE Or
BS_AUTORADIOBUTTON, 30, 160, 110, 24, GethWnd, 0, 0, 0)

    'テキストボックス作成(境界線あり・3Dなし)
    Ret = Api_CreateWindowEx(0, "Static", "TEXT1", WS_CHILD Or WS_VISIBLE Or WS_BORDER Or
ES_CENTER, 20, 210, 130, 24, GethWnd, 0, 0, 0)

    'エディットボックス作成(境界線あり・3Dなし)
    Ret = Api_CreateWindowEx(0, "Edit", "EDIT1", WS_CHILD Or WS_VISIBLE Or WS_BORDER Or
ES_LEFT , 180, 10, 130, 24, GethWnd, 0, 0, 0)

    'エディットボックス作成(複数行入力あり・3Dあり)
    StdStyles = WS_BORDER Or WS_CHILD Or WS_HSCROLL Or WS_VSCROLL Or WS_VISIBLE Or
ES_AUTOHSCROLL Or ES_AUTOVSCROLL Or ES_MULTILINE Or ES_LEFT
    Ret = Api_CreateWindowEx(WS_EX_CLIENTEDGE, "Edit", "EDIT2", StdStyles, 180, 40, 130,
80, GethWnd, 0, 0, 0)

    'コンボボックス作成
    Ret = Api_CreateWindowEx(0, "ComboBox", "COMBO1", WS_CHILD Or WS_VISIBLE Or
CBS_SIMPLE Or CBS_DROPDOWN, 180, 130, 130, 100, GethWnd, 0, 0, 0)
End Sub

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

ウィンドウ(コントロール)をコードで作成(Visual Style)

BUTTON、CHECK、GROUP、EDIT、COMBOの各コントロールをWindowsXPスタイルで作成します。
WindowsXPスタイルは、当然ながらWindowsXPでなければ効果はありません。VB6(FB)についてはサポート外ですので、不具合もあるようです。(最下段参照)

コントロールをコードで作成と全く同じですが、WindowsXP(ビジュアル)スタイルで表示します。
GetSysColor システムのCOLOR取得
CreateWindowEx 新しいウインドウ(コントロール)を作成
DestroyWindow ウインドウ(コントロール)の解放

マニフェストを作成しアプリケーションがビジュアル スタイルを使用できるようにします。
以下のマニフェストファイルをメモ帳などで作成、実行ファイル名と同じにし、同じフォルダに置いてください。
例:本体実行ファイル名を CreateWindowEx.exe とした場合、CreateWindowEx.exe.manifest とします。青色で表している個所は各exeファイルの内容により変わります。

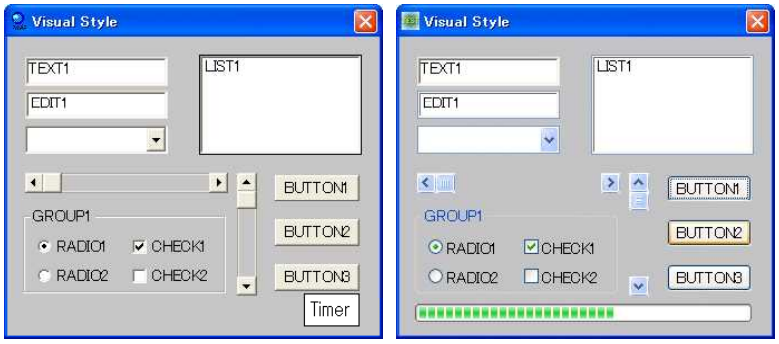


MAINFORMは、可視なし、BUTTON1を貼り付けます。
起動後BUTTON1の表示色(COLOR_BTNFACE)を取得し、MAINFORMをその色で塗りつぶしています。
フォーム作成時(左)・起動時(右)ボタンの角が丸みを帯びている

BUTTON1をクリックすることによりその他のコントロールを作成します。チェックの色（緑）・スクロールバー・コンボボックスなどがビジュアルスタイルに・・・

※F-Basicでフォームに貼り付けたスクロールバーはこんな風になってしまいました。^^;）
プログレスバー部のみ<プログレスバーの作成>のコードを流用しています。

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
<assemblyIdentity
    version="1.0.0.0"
    processorArchitecture="X86"
    name="CompanyName.ProductName.CreateWindowEx.exe"
    type="win32"
/>
<description>コントロールをWindowsXPスタイルで(Visual Style)</description>
<dependency>
    <dependentAssembly>
        <assemblyIdentity
            type="win32"
            name="Microsoft.Windows.Common-Controls"
            version="6.0.0.0"
            processorArchitecture="X86"
            publicKeyToken="6595b64144ccf1df"
            language="*"
        />
    </dependentAssembly>
</dependency>
</assembly>
```



参照
★コントロールをコードで作成<コントロールをコードで作成>
★Windows XP ビジュアル スタイルについて
http://www.microsoft.com/japan/msdn/windows/windowsxp/xptheming.asp#xptheming_topic3
http://support.microsoft.com/kb/309366/ja

```
'=======================================
'= CreateWindowExのテスト
'=    (CreateWindowEx2.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&)

#define COLOR_BTNFACE 15                    '3Dオブジェクトの表面色

' ウィンドウ（コントロール）を作成
Declare Function Api_CreateWindowEx& Lib "user32" Alias "CreateWindowExA" (ByVal
```

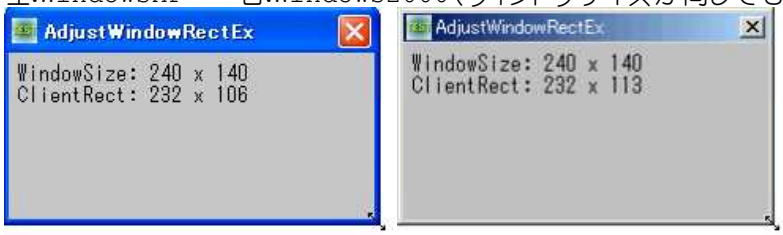
```
ExStyle&, ByVal ClassName$, ByVal WinName$, ByVal Style&, ByVal x&, ByVal y&, ByVal
nWidth&, ByVal nHeight&, ByVal Parent&, ByVal Menu&, ByVal Instance&, ByVal Param&)

' CreateWindowExの解放
Declare Function Api_DestroyWindow& Lib "user32" Alias "DestroyWindow" (ByVal hWnd&)

#define WS_BORDER &H800000                  'フォームの枠線がある
#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_CLIPSIBLINGS &H4000000           '兄弟関係にある子ウィンドウをクリップする
#define WS_EX_WINDOWEDGE &H100              'ウィンドウが盛り上がった縁の境界線を持つように指定
#define WS_EX_OVERLAPPEDWINDOW &H300        'WS_EX_WINDOWEDGEとWS_EX_CLIENTEDGEの組み合せ
#define WS_VISIBLE &H10000000               '可視状態の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 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_EX_CLIENTEDGE &H200

#define BS_PUSHBUTTON &H0                   'プッシュボタン
#define BS_AUTOCHECKBOX &H3                 'チェックボックス
#define BS_AUTORADIOBUTTON &H9              'ラジオボタン
#define BS_GROUPBOX &H7                     'グループボックス

#define ES_AUTOHSCROLL &H80                 '入力範囲を越えたら自動的に水平スクロールする
#define ES_AUTOVSCROLL &H40                 '最後の行で自動的に垂直スクロールをする
#define ES_CENTER 1                         'テキストを水平方向で中央に表示する
#define ES_LEFT 0                           'テキストを左揃えする(デフォルト)
#define ES_LOWERCASE &H10                   '入力文字を小文字にする
#define ES_MULTILINE 4                      '複数行エディットを作成する
#define ES_NOHIDESEL &H100                  'フォーカスを失っても選択範囲を表示する
#define ES_NUMBER &H2000                    '数値入力専用にする
#define ES_OEMCONVERT &H400                 '入力文字をOEM文字セットに変換する
#define ES_PASSWORD &H20                    '入力文字をデフォルトでは*にして表示する
#define ES_READONLY &H800                   '読みとり専用にする(選択してコピーはできる)
#define ES_RIGHT 2                          'テキストを右揃えする
#define ES_UPPERCASE 8                      '入力文字を大文字にする
#define ES_WANTRETURN &H1000                '複数行エディットでEnterキーで改行する

#define CBS_SIMPLE &H1                      '単純なコンボボックス。リストは常にドロップダウンる
#define CBS_DROPDOWN &H2                    'CBS_SIMPLEで、リストはドロップダウンアイコンで表示

Var Shared Button1 As Object

Button1.Attach GetDlgItem("Button1")

'=====================================
'=
'=====================================
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 StdStyles As Long
    Var Ret As Long
```

```
    'コマンドボタン作成
    Ret = Api_CreateWindowEx(0, "Button", "BUTTON2", WS_CHILD Or WS_VISIBLE Or
BS_PUSHBUTTON, 20, 50, 130, 24, GethWnd, 0, 0, 0)

    'グループ作成
    Ret = Api_CreateWindowEx(0, "Button", "GROUP1", WS_CHILD Or WS_VISIBLE Or
BS_GROUPBOX, 20, 90, 130, 110, GethWnd, 0, 0, 0)

    'チェックボタン作成
    Ret = Api_CreateWindowEx(0, "Button", "CHECK1", WS_CHILD Or WS_VISIBLE Or
BS_AUTOCHECKBOX, 30, 120, 110, 24, GethWnd, 0, 0, 0)

    'ラジオボタン作成
    Ret = Api_CreateWindowEx(0, "Button", "RADIO1", WS_CHILD Or WS_VISIBLE Or
BS_AUTORADIOBUTTON, 30, 160, 110, 24, GethWnd, 0, 0, 0)

    'テキストボックス作成(境界線あり・3Dなし)
    Ret = Api_CreateWindowEx(0, "Static", "TEXT1", WS_CHILD Or WS_VISIBLE Or WS_BORDER Or
ES_CENTER, 20, 210, 130, 24, GethWnd, 0, 0, 0)

    'エディットボックス作成(境界線あり・3Dなし)
    Ret = Api_CreateWindowEx(0, "Edit", "EDIT1", WS_CHILD Or WS_VISIBLE Or WS_BORDER Or
ES_LEFT , 180, 10, 130, 24, GethWnd, 0, 0, 0)

    'エディットボックス作成(複数行入力あり・3Dあり)
    StdStyles = WS_BORDER Or WS_CHILD Or WS_HSCROLL Or WS_VSCROLL Or WS_VISIBLE Or
ES_AUTOHSCROLL Or ES_AUTOVSCROLL Or ES_MULTILINE Or ES_LEFT
    Ret = Api_CreateWindowEx(WS_EX_CLIENTEDGE, "Edit", "EDIT2", StdStyles, 180, 40, 130,
80, GethWnd, 0, 0, 0)

    'コンボボックス作成
    Ret = Api_CreateWindowEx(0, "ComboBox", "COMBO1", WS_CHILD Or WS_VISIBLE Or
CBS_SIMPLE Or CBS_DROPDOWN, 180, 130, 130, 100, GethWnd, 0, 0, 0)
End Sub

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

ウィンドウサイズとクライアントサイズ

ウィンドウ(フォーム)サイズとクライアントサイズ(フレーム・タイトルの高さなどを差し引いた領域)の設定
AdjustWindowRectEx クライアントサイズぴったりの大きさのウィンドウを作成
GetClientRect ウィンドウのクライアント領域の座標を取得
GetWindowLong 指定されたウィンドウに関しての情報を取得

左:WindowsXP　　右:Windows2000(ウィンドウサイズが同じでもクライアント領域はこのように異なる)



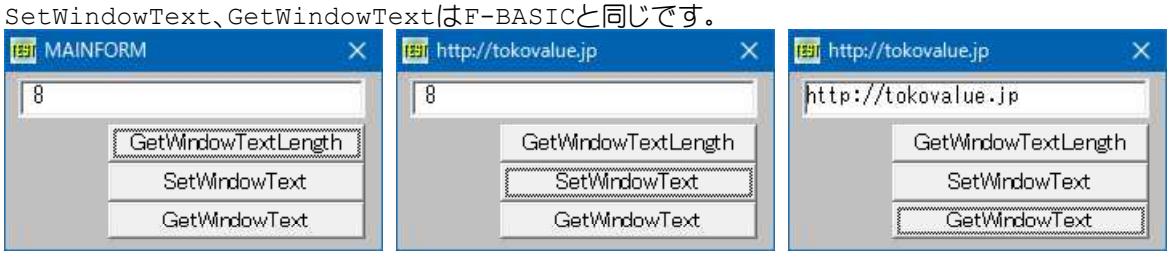

```
'=====================================
'= ウィンドウサイズとクライアントサイズ
'=     (AdjustWindowRectEx.Ex)
'=====================================
#include "Windows.bi"
```

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

' クライアントサイズぴったりの大きさのウィンドウを作成
Declare Function Api_AdjustWindowRectEx& Lib "user32" Alias "AdjustWindowRectEx" (lpRect
As RECT, ByVal dsStyle&, ByVal bMenu&, ByVal dwEsStyle&)

' 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_EXSTYLE (-20)                   '拡張ウィンドウスタイル
#define GWL_STYLE (-16)                     'ウインドウスタイルを取得
Var Shared Text1 As Object

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

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var mWidth As Long
    Var mHeight As Long
    Var rc As RECT
    Var Ret As Long

    rc.Left = 0
    rc.Top = 0
    rc.Right = 232
    rc.Bottom = 106

    Ret = Api_AdjustWindowRectEx(rc, Api_GetWindowLong(GethWnd, GWL_STYLE), 0,
Api_GetWindowLong(GethWnd, GWL_EXSTYLE))

    mWidth = rc.Right - rc.Left
    mHeight = rc.Bottom - rc.Top
    SetWindowSize mWidth, mHeight
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Resize edecl ()
Sub MainForm_Resize()
    Var rc As RECT
    Var Ret As Long

    Ret = Api_GetClientRect(GethWnd, rc)

    Text1.SetWindowText "WindowSize:" & Str$(GetWidth) & " x" & Str$(GetHeight) & Chr$(13,
10) & "ClientRect:" & Str$(rc.Right) & " x" & Str$(rc.Bottom)
End Sub

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

ウィンドウのタイトル文字数を取得

ウィンドウのタイトル文字数を取得します。
GetWindowTextLength ウィンドウのタイトル文字数を取得
SetWindowText ウインドウのタイトルを変更
GetWindowText ウインドウのタイトル文字列を取得
GetActiveWindow アクティブなウィンドウのハンドルを取得

SetWindowText、GetWindowTextはF-BASICと同じです。




```
'=====================================
'= ウィンドウのタイトル文字数を取得
'=    (GetWindowTextLength.bas)
'=====================================
#include "Windows.bi"

' 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を変更
Declare Function Api_SetWindowText& Lib "user32" Alias "SetWindowTextA" (ByVal hWnd&,
ByVal lpString$)

' 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&)

' アクティブなウィンドウのハンドルを取得
Declare Function Api_GetActiveWindow& Lib "user32" Alias "GetActiveWindow" ()

Var Shared Text1 As Object
Var Shared Button1 As object
Var Shared Button2 As object
Var Shared Button3 As object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14
Button3.Attach GetDlgItem("Button3") : Button3.SetFontSize 14

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

    ActiveWnd = Api_GetActiveWindow()
    Ret = Api_GetWindowTextLength(ActiveWnd)
    Text1.SetWindowText Str$(Ret)
End Sub


'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var ActiveWnd As Long
```

```
    Var NewTitle As String
    Var Ret As Long

    NewTitle = "tokovalue.hp.infoseek.co.jp"

    ActiveWnd = Api_GetActiveWindow()
    Ret = Api_SetWindowText(ActiveWnd, NewTitle)
End Sub

'=======================================
'=
'=======================================
Declare Sub Button3_on edecl ()
Sub Button3_on()
    Var ActiveWnd As Long
    Var Buffer As String
    Var Char As Integer

    Buffer = Space$(255)
    ActiveWnd = Api_GetActiveWindow()
    Char = Api_GetWindowText(ActiveWnd, Buffer, 255)
    Text1.SetWindowText Left$(Buffer, Len(Buffer)-1)
End Sub

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

ウインドウテキスト(タイトル)の取得と設定

ウインドウテキスト(タイトル)の取得と変更をします。
F-BASICでのGetWindowTextおよびSetWindowTextと同じです。
GetWindowText ウインドウテキストの取得
SetWindowText ウインドウテキストの設定
GetWindowTextLength ウインドウタイトルバーの文字列の長さを取得


```
'=======================================
'= ウインドウテキスト(タイトル)の取得と設定
'=    (GetWindowtext.bas)
'=======================================
#include "Windows.bi"

' 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&)

' 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$)

' 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&)
```

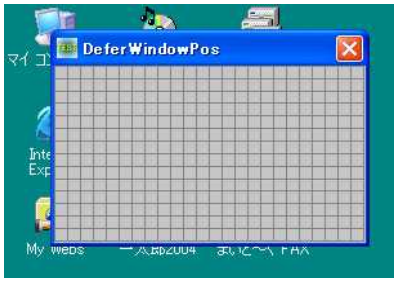
```
Var Shared Edit1 As Object
Edit1.Attach GetDlgItem("Edit1")

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

    MyStr = String$(Api_GetWindowTextLength(GethWnd) + 1, Chr$(0))
    Ret = Api_GetWindowText(GethWnd, MyStr, Len(MyStr))
    Edit1.SetWindowText MyStr
End Sub

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

    MyStr = Edit1.GetWindowText
    Ret = Api_SetWindowText(GethWnd, MyStr)
    Edit1.SetWindowText ""
End Sub

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

ウィンドウの位置とサイズを更新

AdjustWindowRect クライアント範囲から必要なウィンドウのサイズを得る
BeginDeferWindowPos ウィンドウ位置構造体を作成
DeferWindowPos 複数のウィンドウ位置構造体を更新する
EndDeferWindowPos 複数のウィンドウの位置とサイズを更新する

フォーム設計時のサイズ 240×140

設計時のフォームと重ねてみる




```
    R.Left = 40         'フォーム左位置40ドット目
    R.Top = 50          '    〃   上位置50ドット目
    R.RIGHT = 280       '    〃   右位置280ドット目
    R.Bottom = 190      '    〃   下位置190ドット目
```

寸法の把握を容易にするため10ドット毎に線を引いています。

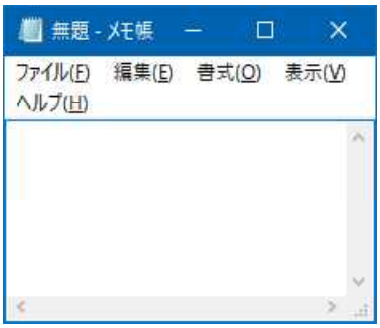
ウインドウサイズについての参考URL
http://www.arcpit.co.jp/winapi/api_02/ap020201.htm

```
'====================================
'= 位置構造体によるウィンドウの位置とサイズを更新
'=    (DeferWindowPos.bas)
'====================================
#include "Windows.bi"

#define WS_BORDER &H800000                  'フォームの枠線がある
#define WS_DLGFRAME &H400000                'ダイアログボックスで一般的に使われるスタイルの境界を
                                            持つウィンドウを作成する
#define WS_THICKFRAME &H40000               'サイズ変更境界を持つウィンドウを作成する
#define WS_CAPTION &HC00000                 'WS_BORDER Or WS_DLGFRAME
#define HWND_BOTTOM 1                       'ウインドウをウインドウリストの一番下に配置する
#define HWND_NOTOPMOST -2                   'ウインドウをウインドウリストの一番上に配置する
#define HWND_TOP 0                          'ウインドウをZオーダーの一番上に配置する
#define HWND_TOPMOST -1                     'ウインドウをウインドウリストの一番上に配置する
#define SWP_SHOWWINDOW &H40                 'ウインドウを表示する

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

' クライアント範囲から必要なウィンドウのサイズを得る
Declare Function Api_AdjustWindowRect& Lib "user32" Alias "AdjustWindowRect" (lpRect As
RECT, ByVal dwStyle&, ByVal bMenu&)

' ウィンドウ位置構造体を作成
Declare Function Api_BeginDeferWindowPos& Lib "user32" Alias "BeginDeferWindowPos"
(ByVal nNumWindows&)

' 複数のウィンドウ位置構造体を更新する
Declare Function Api_DeferWindowPos& Lib "user32" Alias "DeferWindowPos" (ByVal
hWinPosInfo&, ByVal hWnd&, ByVal hWndInsertAfter&, ByVal x&, ByVal y&, ByVal cx&, ByVal
cy&, ByVal wFlags&)

' 複数のウィンドウの位置とサイズを更新する
Declare Function Api_EndDeferWindowPos& Lib "user32" Alias "EndDeferWindowPos" (ByVal
hWinPosInfo&)

Var Shared Mainform As Object
Mainform.Attach GethWnd

'====================================
'=
'====================================
Declare Sub Mainform_Start edecl ()
Sub Mainform_Start()
    Var R As RECT
    Var hDWP As Long

    R.Left = 40
    R.Top = 50
    R.RIGHT = 280
    R.Bottom = 190

    Ret = Api_AdjustWindowRect(R, WS_THICKFRAME Or WS_CAPTION, False)
    hDWP = Api_BeginDeferWindowPos(1)
    Ret = Api_DeferWindowPos(hDWP, GethWnd, HWND_TOP, R.Left, R.Top, R.Right - R.Left,
R.Bottom - R.Top, SWP_SHOWWINDOW)
    Ret = Api_EndDeferWindowPos(hDWP)

    Mainform.ShowWindow -1

    For x = 0 To 240 step 10
        line(x, 0)-(x, 140),,1
    Next

    For y = 0 To 140 step 10
```

```
        line(0, y)-(240, y),,1
    Next
End Sub

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

ウィンドウ位置・サイズを設定（自分自身・外部）

メモ帳を起動し、その位置およびサイズ、また自分自身の位置およびサイズを設定します。
DeferWindowPos 複数ウィンドウ位置構造体に、ウィンドウの移動先の位置情報を格納
BeginDeferWindowPos 複数ウィンドウ位置構造体にメモリーを割り当て、この構造体のハンドルを返す
EndDeferWindowPos 複数のウィンドウの位置やサイズを一斉に更新
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



図は、起動直後の状態。数値を変えて「実行」をクリックし位置およびサイズの確認をします。
自身サイズのサイズは「実行」ボタンが隠れない程度に設定してください。

```
'=====================================
'= ウィンドウ位置およびサイズを設定
'=    (DeferWindowPos2.bas)
'=====================================
#include "Windows.bi"

' 複数ウィンドウ位置構造体に、ウィンドウの移動先の位置情報を格納
Declare Function Api_DeferWindowPos& Lib "user32" Alias "DeferWindowPos" (ByVal
hWinPosInfo&, ByVal hWnd&, ByVal hWndInsertAfter&, ByVal x&, ByVal y&, ByVal cx&, ByVal
cy&, ByVal wFlags&)

' 複数ウィンドウ位置構造体にメモリーを割り当て、この構造体のハンドルを返す
Declare Function Api_BeginDeferWindowPos& Lib "user32" Alias "BeginDeferWindowPos"
(ByVal nNumWindows&)

' 複数のウィンドウの位置やサイズを一斉に更新
Declare Function Api_EndDeferWindowPos& Lib "user32" Alias "EndDeferWindowPos" (ByVal
hWinPosInfo&)

' 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 HWND_TOP 0                  'ウィンドウをZオーダーの一番上に配置する
#define HWND_BOTTOM 1               'ウィンドウをウィンドウリストの一番下に配置する
#define SWP_SHOWWINDOW &H40         'ウィンドウを表示する
#define vbNormalFocus 1             '開いたアプリケーションはフォーカスを持ち、前回起動したサイズと位置に復元される
```

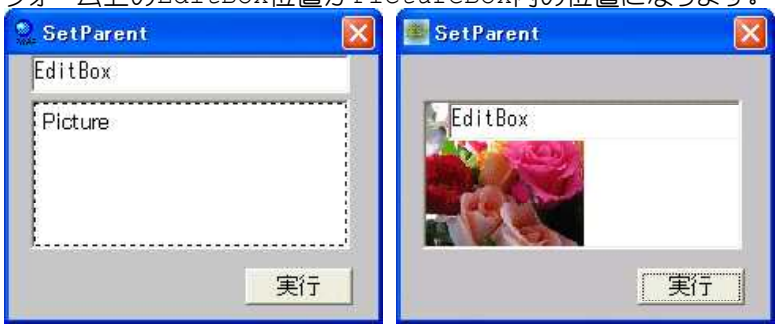
```
#define WM_CLOSE &H10                          'ウィンドウ或いはアプリケーションをクローズされた
#define vbNullString ByVal 0                  '値0の文字列。値0を持つ文字列。空文字列ではない

Var Shared Text(3) As Object
Var Shared Edit(7) As Object
Var Shared Button1 As Object
Var Shared Button2 As Object

For i = 0 To 7
    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
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14

Var Shared hWnd As Long

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

    'コマンドボタンを無効に設定
    Button1.EnableWindow 0

    'メモ帳を起動
    Shell "Notepad.exe", , 5
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var ClassName As String
    Var pWnd As Long
    Var fPos(7) As Integer
    Var Ret As Long

    For i = 0 To 7
        fPos(i) = Val(Edit(i).GetWindowText)
    Next

    'クラス名でウィンドウハンドルを取得
    ClassName = "Notepad"
    hWnd = Api_FindWindow(ClassName, vbNullString)

    'ウィンドウハンドルを取得できたとき
    If hWnd <> 0 Then

        '複数ウィンドウ位置構造体のハンドルを取得
        pWnd = Api_BeginDeferWindowPos(2)

        'フォームの位置とサイズを設定
        pWnd = Api_DeferWindowPos(pWnd, GethWnd, HWND_BOTTOM, fPos(0), fPos(1), fPos(2),
fPos(3), SWP_SHOWWINDOW)


        'メモ帳の位置とサイズを設定
        pWnd = Api_DeferWindowPos(pWnd, hWnd, HWND_TOP, fPos(4), fPos(5), fPos(6),
fPos(7), SWP_SHOWWINDOW)

        'ウィンドウの位置とサイズを一斉に変更
        Ret = Api_EndDeferWindowPos(pWnd)
    End If
```

```
End Sub

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

    Ret = Api_SendMessage(hWnd, WM_CLOSE, 0, 0)
End Sub

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

ウインドウの親子関係

SetParent ウインドウの親子関係を設定します。

例ではPicture1を親、Edit1を子としています。「実行ボタン」によりEdit1をPicture1の中に入れています。
フォーム上のEditBox位置がPictureBox内の位置になります。

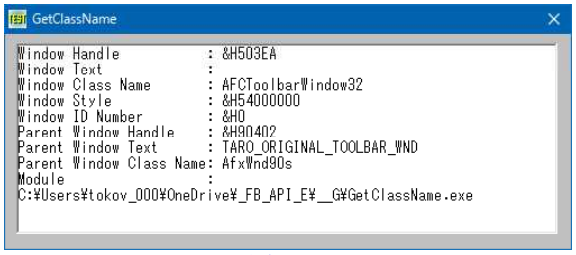



```
'=====================================
'= ウインドウの親子関係
'=    (SetParent.bas)
'=====================================
#include "Windows.bi"

' ウインドウの親子関係を設定
Declare Function Api_SetParent& Lib "user32" Alias "SetParent" (ByVal hWndChild&, ByVal
hWndNewParent&)

Var Shared Edit1 As Object
Var Shared Picture1 As Object
Var Shared Bitmap As Object
BitmapObject Bitmap

Edit1.Attach GetDlgItem("Edit1")
```

```
Picture1.Attach GetDlgItem("Picture1")

'=====================================
'=
'=====================================
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 Ret As Long

    Ret = Api_SetParent(Edit1.GethWnd, Picture1.GethWnd)
End Sub

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

ウィンドウのクラス名や属性を取得

WindowFromPoint 指定座標位置にあるウィンドウハンドルを取得
GetWindowWord ウィンドウに関連付けた補足データ領域からワード値を取得 (GetWindowLongを推奨)
GetCursorPos 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
GetModuleFileName ロードされている実行モジュールのフルパスを取得
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のハンドルを取得
GetClassName ウィンドウのクラス名を取得
GetWindowText ウィンドウのタイトル文字列を取得



参照(F-Basic用に書換)
http://support.microsoft.com/?kbid=112649

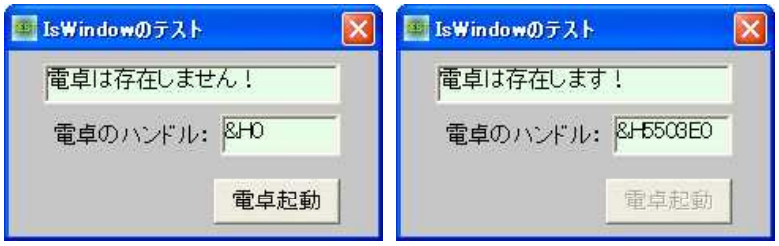
```
'=====================================
'= ウィンドウのクラス名や属性を取得
'=    (GetClassName.bas)
'=====================================
#include "Windows.bi"

Type POINTAPI
    x   As Long
    y   As Long
End Type

' 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&)

' ウィンドウに関連付けて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&)

' 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)

' 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&)

' 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 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Timer1.SetInterval 50
    Timer1.Enable -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var pa 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(pa)                  'カーソル位置を取得
    ptx = pa.x
    pty = pa.y

    'カーソルの位置しているウィンドウを取得
    hWndOver = Api_WindowFromPoint(ptx, pty)

    txt = ""
    If hWndOver <> hWndLast Then                '変更があったら表示する
        hWndLast = hWndOver                     '変更の保存
        txt = txt & "Window Handle          : &&H" & Hex$(hWndOver) & vbCrLf
                                                'ウィンドウハンドルの表示
        Ret = Api_GetWindowText(hWndOver, sWindowText, 100)'ウィンドウテキスト
        txt = txt & "Window Text            : " & Left$(sWindowText, Ret) & vbCrLf

        Ret = Api_GetClassName(hWndOver, sClassName, 100)       'ウィンドウクラス
        txt = txt & "Window Class Name      : " & Left$(sClassName, Ret) & vbCrLf

        lWindowStyle = Api_GetWindowLong(hWndOver, GWL_STYLE)   'ウィンドウスタイル
        txt = txt & "Window Style           : &&H" & Hex$(lWindowStyle) & vbCrLf

        '親ウィンドウハンドルの取得
        hWndParent = Api_GetParent(hWndOver)

        If hWndParent <> 0 Then

            'ウィンドウ ID の取得
            wID = Api_GetWindowWord(hWndOver, GWW_ID)
            txt = txt & "Window ID Number       : &&H" & Hex$(wID) & vbCrLf
            txt = txt & "Parent Window Handle   : &&H" & Hex$(hWndParent) & vbCrLf

            '親ウィンドウテキストの取得
            Ret = Api_GetWindowText(hWndParent, sParentWindowText, 100)
            txt = txt & "Parent Window Text     : " & Left$(sParentWindowText, Ret) & vbCrLf

            '親ウィンドウクラス名の取得
            Ret = Api_GetClassName(hWndParent, sParentClassName, 100)
            txt = txt & "Parent Window Class Name: " & Left$(sParentClassName, Ret) & vbCrLf
        Else

            '親ウィンドウがないとき
            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の取得
        hInstance = Api_GetWindowWord(hWndOver, GWW_HINSTANCE)

        'モジュール名の取得
        Ret = Api_GetModuleFileName(GethInst, sModuleFileName, 100)
        txt = txt & "Module                 : " & Left$(sModuleFileName, Ret) & vbCrLf
        Text1.SetWindowText txt
    End If
End Sub

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

ウインドウの存在を確認

IsWindow 指定のウインドウが、存在するか調べる
FindWindow クラス名とウィンドウ名が指定された文字列と一致するトップ レベル ウィンドウのハンドルを取得




```
'=======================================
'= ウインドウの存在を確認
'=    (IsWindow.bas)
'=======================================
#include "Windows.bi"

' 指定されたウィンドウ ハンドルが既存のウィンドウを識別しているどうかを判断
Declare Function Api_IsWindow& Lib "user32" Alias "IsWindow" (ByVal hWnd&)

' 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$,
ByVal lpWindowName$)

Var Shared Text(2) As Object
Var Shared Button1 As Object
Var Shared Timer1 As Object

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
Timer1.Attach GetDlgItem("Timer1")

Var Shared hWnd As Long

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

    Shell "Calc.exe"

    Timer1.SetInterval 100
    Timer1.Enable -1
End Sub

'=======================================
'=
'=======================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var ClassName$ As String
    Var Caption$ As String
    Var Ret As Long

    ClassName$ = "SciCalc"
    Caption$ = "電卓"

    '電卓のハンドル取得
    hWnd = Api_FindWindow(ClassName$, ByVal 0)
    Text(1).SetWindowText "&&H" & Hex$(hWnd)
```

タイマーで電卓の存在有無を確認し、存在する場合そのハンドル値を表示します。

```
    'ウインドウの存在確認
    Ret = Api_IsWindow(hWnd)

    If Ret = 0 Then
        Text(0).SetWindowText "電卓は存在しません！"
        Button1.EnableWindow -1
    Else
        Text(0).SetWindowText "電卓は存在します！"
        Button1.EnableWindow 0
    End If
End Sub

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

ウインドウの透明度を変更する

ウインドウ(フォーム)の透明度をスクロールバーで変更しています。例では、カラーキーの設定はしていません。
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
SetLayeredWindowAttributes 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

起動時Alpha値最大255

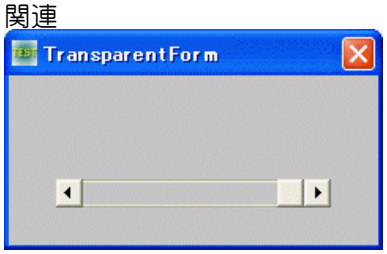
Alpha値218

Alpha値160

Alpha値110

Alpha値72(スクロールバーの最小値は25に設定しています)

関連

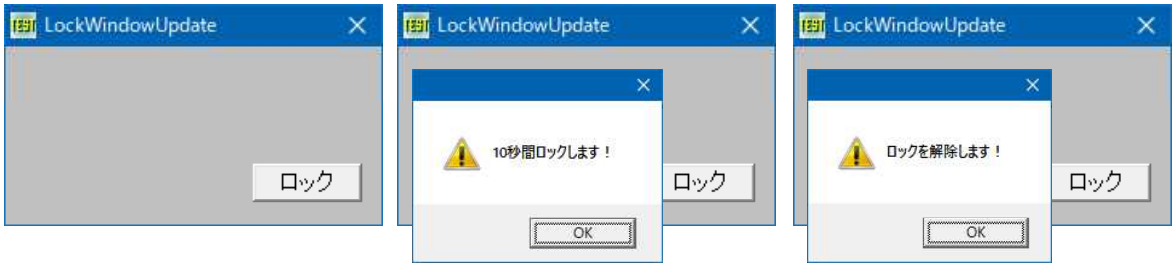



```
'=====================================
'= ウインドウの透明度変更
'=    (SetLayeredWindowAttributes.bas)
'=====================================
#include "Windows.bi"

#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 GWL_EXSTYLE -20                 '拡張ウィンドウスタイル
#define WS_EX_LAYERED &H80000           '透明なウィンドウ属性(Windows2000以上)
```

```
' 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&)

' 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 As Byte, ByVal dwFlags&)

Var Shared Text1 As Object
Var Shared HScroll1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
HScroll1.Attach GetDlgItem("HScroll1")

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

    HScroll1.SetScrollRange 25,255
    HScroll1.SetScrollStep 5, 10
    HScroll1.SetScrollPos 255

    Text1.SetWindowtext format$(255, "Alpha:###")

    Ret = Api_GetWindowLong(GethWnd, GWL_EXSTYLE)
    Ret = Ret Or WS_EX_LAYERED
    Ret = Api_SetWindowLong(GethWnd, GWL_EXSTYLE, Ret)
End Sub

'=======================================
'=
'=======================================
Declare Sub HScroll1_Change edecl ()
Sub HScroll1_Change()
    Var Alpha As Byte
    Var Ret As Long

    '半透明化(25 ～ 255)
    Alpha = HScroll1.GetScrollPos
    Text1.SetWindowtext format$(Alpha, "Alpha:###")

    Ret = Api_SetLayeredWindowAttributes(GethWnd, 0, Alpha, LWA_ALPHA)
End Sub

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

ウィンドウの描画をロック・解除

LockWindowUpdate 指定されたウィンドウ内での描画を無効または有効にする
GetDesktopWindow Windowsのデスクトップウィンドウを識別



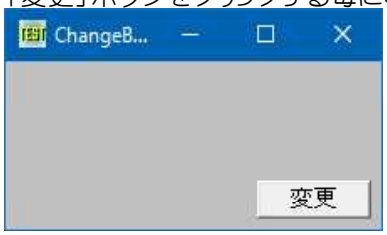

```
'=====================================
'= ウィンドウの描画をロック・解除
'=    (LockWindowUpdate.bas)
'=====================================
#include "Windows.bi"

' 指定されたウィンドウ内での描画を無効または有効にする
Declare Function Api_LockWindowUpdate& Lib "user32" Alias "LockWindowUpdate" (ByVal
hwndLock&)

' 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" ()

Var Shared Timer1 As Object
Var Shared Button1 As Object

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

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Timer1.SetInterval 1000
    Timer1.Enable 0
End Sub

'=====================================
'= ロック実行
'=====================================
Declare Sub Freeze ()
Sub Freeze()
    Var Ret As Long

    Ret = Api_LockWindowUpdate(Api_GetDesktopWindow)
End Sub

'=====================================
'= ロック解除
'=====================================
Declare Sub Defrost ()
Sub Defrost()
    Var Ret As Long

    Ret = Api_LockWindowUpdate(0)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    A% = MessageBox("", "10秒間ロックします！", 0, 2)
    EnableWindow 0

    Timer1.Enable -1
End Sub
```

```
'====================================
'=
'====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Timer1.Enable 0

    Defrost

    A% = MessageBox("", "ロックを解除します！", 0, 2)

    EnableWindow -1
End Sub

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

ウィンドウのボーダースタイルを変更

ウィンドウ(Form)のボーダースタイルを変更します。
GetWindowLong 指定されたウィンドウに対しての情報を取得
SetWindowLong 指定されたウィンドウの属性を変更

「変更」ボタンをクリックする毎に、BorderStyleを変更しています。



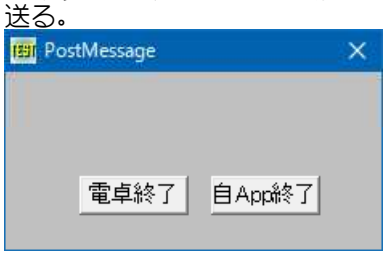

```
'====================================
'= ウィンドウのボーダースタイルを変更
'=    (ChangeBorderStyle.bas)
'====================================
#include "Windows.bi"

#define GWL_STYLE -16                       'アプリケーションのインスタンスハンドル
#define GWL_EXSTYLE -20                     '拡張ウィンドウスタイル
#define WS_THICKFRAME &H40000               'サイズ変更境界を持つウィンドウを作成する
#define WS_MAXIMIZEBOX &H10000              '最大化ボタンを持つウィンドウを作成する
#define WS_MINIMIZEBOX &H20000              '最小化ボタンを持つウィンドウを作成する
#define WS_EX_TOOLWINDOW &H80               'ツールウィンドウを作成

' 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&)

#define SWP_NOSIZE &H1                      'ウインドウの現在のサイズを保持する
#define SWP_NOMOVE &H2                      'ウインドウの現在位置を保持する
#define SWP_NOZORDER &H4                    'ウインドウリスト内での現在位置を保持する
#define SWP_NOREDRAW &H8                    'ウインドウを自動的に再描画しない
#define SWP_NOACTIVATE &H10                 'ウインドウをアクティブにしない
```

```
#define SWP_FRAMECHANGED &H20                'ウインドウのサイズ変更中でなくてもWM_NCCALCSIZEを送る
#define SWP_SHOWWINDOW &H40                  'ウインドウを表示する
#define SWP_HIDEWINDOW &H80                  'ウインドウを隠す
#define SWP_NOCOPYBITS &H100                 'クライアント領域の内容をクリアする
#define SWP_NOOWNERZORDER &H200              'オーナーウインドウのZオーダーは変えない
#define SWP_DRAWFRAME &H20                   '再描画のときウインドウを囲む枠も描画
#define SWP_NOREPOSITION SWP_NOOWNERZORDER

#define HWND_TOP 0                           'ウインドウをZオーダーの一番上に配置する
#define HWND_BOTTOM 1                        'ウインドウをウインドウリストの一番下に配置する
#define HWND_TOPMOST -1                      'ウインドウをウインドウリストの一番上に配置する
#define HWND_NOTOPMOST -2                    'ウインドウをウインドウリストの一番上に配置する

' ウィンドウのサイズ、位置、および Z オーダーを設定。(ウィンドウの重なり順のことを「Zオーダー」といいZオーダーのトップに置くと一番手前に表示される)
Declare Function Api_SetWindowPos& Lib "user32" Alias "SetWindowPos" (ByVal hWnd&, ByVal hWndInsertAfter&, ByVal X&, ByVal Y&, ByVal CX&, ByVal CY&, ByVal uFlags&)

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

Var Shared FLG As Integer

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Style1 As Integer
    Var Style2 As Integer
    Var Ret As Long

    FLG = FLG + 1 : If FLG > 2 Then FLG = 0

    Select Case FLG
        Case 0
            Style1 = True
        Case 1
            Style2 = True
        Case Else
            Style1 = True
            Style2 = True
    End Select

    If Style1 Then
        Ret = Api_SetWindowLong(GethWnd, GWL_STYLE, Api_GetWindowLong(GethWnd, GWL_STYLE) xor (WS_THICKFRAME Or WS_MINIMIZEBOX Or WS_MAXIMIZEBOX))
    End If

    If Style2 Then
        Ret = Api_SetWindowLong(GethWnd, GWL_EXSTYLE, Api_GetWindowLong(GethWnd, GWL_EXSTYLE) xor WS_EX_TOOLWINDOW)
    End If

    Ret = Api_SetWindowPos(GethWnd, 0, 0, 0, 0, 0, SWP_NOMOVE Or SWP_NOSIZE Or SWP_NOZORDER Or SWP_FRAMECHANGED)
End Sub

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

ウィンドウのメッセージキューにメッセージを送る

ウィンドウのメッセージキューにメッセージをおくります。
PostMessage 指定のウィンドウにメッセージを送る
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
GetModuleFileName ロードされている事項モジュールのフルパス名を取得

左ボタン:電卓を見つけた場合そのウィンドウにCLOSE命令を送る。　　右ボタン:自アプリケーションにCLOSE命令を送る。




```
'=====================================
'= ウィンドウのメッセージキューにメッセージを送る
'=    (PostMessage.bas)
'=====================================
#include "Windows.bi"

' 指定のウィンドウのメッセージキューにメッセージを送る
Declare Function Api_PostMessage& Lib "user32" Alias "PostMessageA" (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_FindWindow& Lib "user32" Alias "FindWindowA" (ByVal lpClassName$,
ByVal lpWindowName$)

' 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&)

#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

Var Shared Button1 As Object
Var Shared Button2 As Object

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

'=====================================
'= 電卓が見つかった場合、電卓を終了
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hWnd As Long
    Var Ret As Long

    hWnd = Api_FindWindow(ByVal 0, "電卓")
    If hWnd <> 0 Then
        A% = MessageBox(GetWindowText, "電卓を終了しますか？", 1, 1)
    else
        A% = MessageBox(GetWindowText, "電卓は見つかりません！", 0, 2)
        Exit Sub
    End If
    If A% = 1 Then Exit Sub

    Ret = Api_PostMessage(hWnd, WM_CLOSE, 0, ByVal 0)
End Sub

'=====================================
'= 自アプリケーションを終了
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
```

```
    Var Ret As Long
    Var ExePath As String * 128

    Ret = Api_GetModuleFileName(GethInst, ExePath, Len(ExePath))

    A% = MessageBox(ExePath, "自アプリケーションを終了しますか？", 1, 1)
    If A% = 1 Then Exit Sub

    Ret = Api_PostMessage(GethWnd, WM_CLOSE, 0, ByVal 0)
End Sub

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

ウィンドウへの入力の有効無効を設定

起動した電卓への入力を有効化･無効化の設定をします。
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のハンドルを返す
EnableWindow ウィンドウへの入力可能不可能を設定

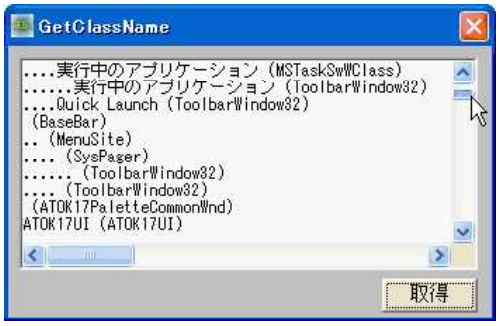
例では、電卓を起動し入力の有効化･無効化を設定しています。

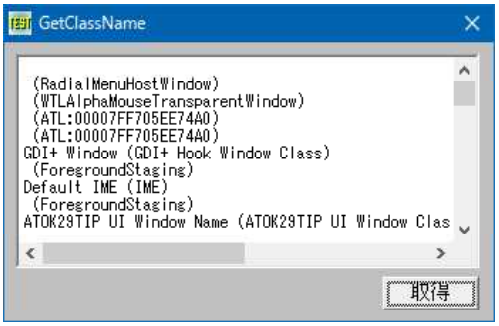

```
'=====================================
'= ウィンドウへの入力の有効無効を設定
'=    (EnableWindow.bas)
'=====================================
#include "Windows.bi"

' 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$, ByVal lpWindowName$)

' ウィンドウへの入力可能不可能を設定
Declare Function Api_EnableWindow& Lib "user32" Alias "EnableWindow" (ByVal hWnd&, ByVal bEnable&)

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

Var Shared Index As Integer

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

```
Declare Sub WindowState edecl ()
Sub WindowState()
    Var ClassName As String
    Var hWnd As Long
    Var Ret As Long

    'クラス名でウィンドウハンドルを取得
    ClassName = "SciCalc"
    hWnd = Api_FindWindow(ClassName, ByVal 0)

    'ウィンドウハンドルを取得できたときは
    If hWnd <> 0 Then

        'ウィンドウへの入力の有効無効を設定
       Ret = Api_EnableWindow(hWnd, Index)
    End If
End Sub

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

    'コマンドボタンを無効に設定
    Button1.EnableWindow 0

    '電卓を起動
    Shell "Calc.exe",, 5
End Sub

'=======================================
'=
'=======================================
Declare Sub Radio1_on edecl ()
Sub Radio1_on()
    Index = 1
    WindowState
End Sub

'=======================================
'=
'=======================================
Declare Sub Radio2_on edecl ()
Sub Radio2_on()
    Index = 0
    WindowState
End Sub

'=======================================
'=
'=======================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Radio1_on
End Sub

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

ウィンドウ名とクラス階層一覧を取得

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
GetWindowText ウィンドウのタイトル文字列を取得
GetTopWindow トップレベルの子ウィンドウを取得
GetClassName ウィンドウのクラス名を取得



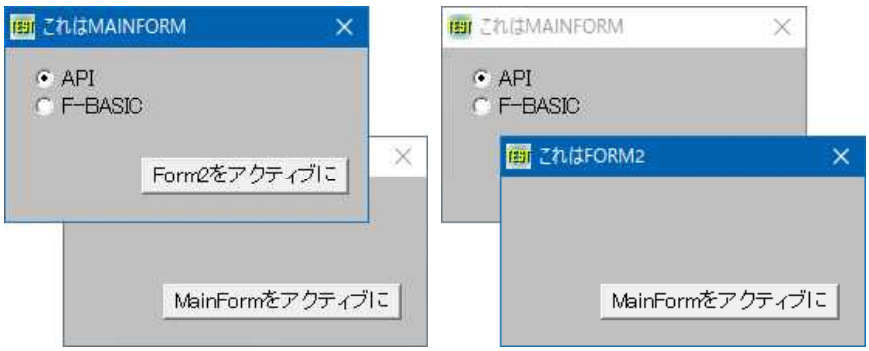



```
'=======================================
'= ウィンドウ名とクラスの階層一覧を取得
'=    (GetClassName2.bas)
'=======================================
#include "Windows.bi"

#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_HWNDNEXT 2 '基準となるウィンドウの次のウィンドウを検索
#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)

' 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&)

' トップレベルの子ウィンドウを取得
Declare Function Api_GetTopWindow& Lib "user32" Alias "GetTopWindow" (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&)

Var Shared Edit1 As Object
Var Shared Button1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 12
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=======================================
'=
'=======================================
Declare Sub AddChildWindows(ByVal hwndParent As Long, ByVal Level As Long)
Sub AddChildWindows(ByVal hwndParent As Long, ByVal Level As Long)
    Var txt As String
    Var WT As String
    Var CN As String
    Var Length As Long
    Var hWnd As Long

    If Level = 0 Then
        hWnd = hwndParent
    Else
        hWnd = Api_GetWindow(hwndParent, GW_CHILD)
    End If

    Do While hWnd <> 0
        WT = Space$(256)
        Length = Api_GetWindowText(hWnd, WT, 255)
```

```
        WT = Left$(WT, Length)

        CN = Space$(256)
        Length = Api_GetClassName(hWnd, CN, 255)
        CN = Left$(CN, Length)

        Edit1.SetWindowText Edit1.GetWindowText & vbCrLf & String$(2 * Level, ".") & WT & "
(" & CN & ")"
        AddChildWindows hWnd, Level + 1
        hWnd = Api_GetWindow(hWnd, GW_HWNDNEXT)
    Loop
End Sub

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

    hWnd = Api_GetTopWindow(0)
    If hWnd <> 0 Then
        AddChildWindows hWnd, 0
    End If
End Sub

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

ウィンドウをアクティブにする

MainFormとForm2を相手方のボタンによりアクティブにします。
F-BASICのSetActiveWindowと全く同じ動作をします。従って、API・F-BASICのどちらをチェックしても同じことです。(^^;;)
SetActiveWindow ウインドウをアクティブに

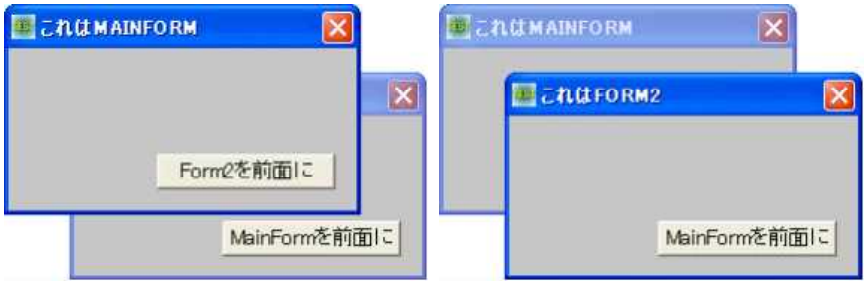



```
'=======================================
'=  ウィンドウを最前面に表示する
'=    (SetActiveWindow.bas)
'=======================================
#include "Windows.bi"

' 指定のウィンドウをアクティブにする
Declare Function Api_SetActiveWindow& Lib "user32" Alias "SetActiveWindow" (ByVal hWnd&)

Var Shared MainForm As Object
Var Shared Form2 As Object
```

```
MainForm.Attach GethWnd

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Form2.CreateWindow "Form2", -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 Ret As Long

    If Index = 0 Then
        Ret = Api_SetActiveWindow(Form2.GethWnd)
    Else
        Form2.SetActiveWindow
    End If
End Sub

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

    If Index = 0 Then
        Ret = Api_SetActiveWindow(MainForm.GethWnd)
    Else
        MainForm.SetActiveWindow
    End If
End Sub

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

ウインドウを最前面に表示する(Ⅰ)

BringWindowToTop ウインドウを一時的に最前面に表示



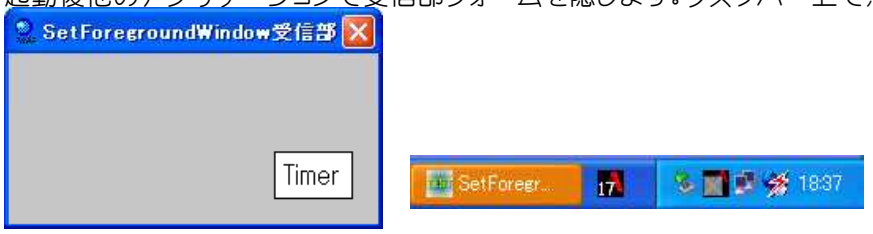

```
'=====================================
'=  ウィンドウを最前面に表示する
'=====================================
#include "Windows.bi"

' ウインドウを一時的に最前面に表示
Declare Function Api_BringWindowToTop& Lib "user32" Alias "BringWindowToTop" (ByVal
hWnd&)
Var Shared MainForm As Object
Var Shared Form2 As Object
MainForm.Attach GethWnd

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Form2.CreateWindow "Form2",-1
End Sub

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

    Ret = Api_BringWindowToTop(Form2.GethWnd)
End Sub

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

    Ret = Api_BringWindowToTop(MainForm.GethWnd)
End Sub

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

ウィンドウを最前面に表示(II)

異なるアプリケーションを最前面に表示させます。(Windows2000/XP)
送信部
GetProp ウィンドウに関するプロパティを取得
FindWindowクラス名またはキャプションを与えてウィンドウのハンドルを取得
AllowSetForegroundWindow 指定したプロセスにフォアグラウンドウィンドウを設定

受信部
SetForegroundWindow フォアグラウンドウィンドウの設定
SetProp ウィンドウに関するプロパティを設定
GetCurrentProcessId 現在のプロセスのプロセスIDを取得

タイマーを貼り付けます。
起動後他のアプリケーションで受信部フォームを隠します。タスクバー上で点滅し起動の確認ができています。



送信部のボタンをクリックすることにより受信部フォームが最前面に表示されます。

※Windows98では受信部フォームが常時最前面に表示されます。

```
'=======================================
'= 指定したウィンドウを最前面に(送信部:Window2000/XP)
'= SetForegroundWindowRecとセット
'=    (SetForegroundWindowSend.bas)
'=======================================
#include "Windows.bi"

' ウィンドウに関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_FindWindow& Lib "user32" Alias "FindWindowA" (ByVal lpClassName$,
ByVal lpWindowName$)

' 指定したプロセスにフォアグラウンドウィンドウを設定
Declare Function Api_AllowSetForegroundWindow& Lib "user32" Alias
"AllowSetForegroundWindow" (ByVal dwProcessId&)

'=======================================
'=
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hWnd As Long
    Var hProcessID As Long
    Var Ret As Long
```

```
    'キャプションが"SetForegroundWindow受信部"であるウィンドウのハンドルを取得
    hWnd = Api_FindWindow(ByVal 0, "SetForegroundWindow受信部")
    If hWnd = 0 Then
        A% = MessageBox("", "目的ウィンドウは見つかりません！", 0, 2)
        Exit Sub
    End If

    hProcessID = Api_GetProp(hWnd, "ProcessID")
    Ret = Api_AllowSetForegroundWindow(hProcessID)
End Sub

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

```
'=====================================
'= 指定したウィンドウを最前面に(受信部:Window2000/XP)
'= SetForegroundWindowSend とセット
'=    (SetForegroundWindowRec.bas)
'=====================================
#include "Windows.bi"

' フォアグラウンドウィンドウの設定
Declare Function Api_SetForegroundWindow& Lib "user32" Alias "SetForegroundWindow"
(ByVal hWnd&)

' 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&)

' 現在のプロセスのプロセス ID を取得
Declare Function Api_GetCurrentProcessId& Lib "kernel32" Alias "GetCurrentProcessId" ()

Var Shared Timer1 As Object
Timer1.Attach GetDlgItem("Timer1")

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

    SetWindowText "SetForegroundWindow受信部"

    Ret = Api_SetProp(GethWnd, "ProcessID", Api_GetCurrentProcessId)

    Timer1.SetInterval 10
    Timer1.Enable -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var Ret As Long

    Ret = Api_SetForegroundWindow(GethWnd)
End Sub

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

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

ウィンドウを最前面に表示（III）

ウィンドウを最前面に表示します。SetForegroundWindowはWindows95、およびWindows98では正常に機能しますが、Windows2000、WindowsXPにおいてはタスクバーで点滅し（仕様:※注参照）、ウィンドウが隠れてしまいます。（IV参照）
（III）では、AttachThreadInput、Timerを使って強制的に全面表示させています。

GetForegroundWindow ユーザーが操作中のウィンドウを取得
SetForegroundWindow 指定されたウィンドウをフォアグラウンドにしアクティブにする
GetWindowThreadProcessId ウィンドウのプロセスIDとスレッドIDを取得
AttachThreadInput 別スレッドの子ウィンドウにフォーカスをセット
SystemParametersInfo システム全体に関するパラメータを取得・設定

※注
SetForegroundWindow
旧バージョンからの変更により、アプリケーションは、ユーザーが他のウィンドウで作業しているときに強制的にフォアグラウンドウィンドウを設定することはできなくなりました。その代わりに、SetForegroundWindow 関数は、ウィンドウをアクティブにし（SetActiveWindow 関数、FlashWindowEx 関数を呼び出してユーザーに通知します。

```
'=====================================
'= ウィンドウを最前面に（III）
'=    (SetForegroundWindow2.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" ()

' 指定されたウィンドウを作成したスレッドをフォアグラウンドにし、そのウィンドウをアクティブにする
Declare Function Api_SetForegroundWindow& Lib "user32" Alias "SetForegroundWindow"
(ByVal hWnd&)

' 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&)

' 別スレッドの子ウィンドウにフォーカスをセット
Declare Function Api_AttachThreadInput& Lib "user32" Alias "AttachThreadInput" (ByVal
idAttach&, ByVal idAttachTo&, ByVal fAttach&)

' 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 uAction&, ByVal uParam&, ByRef lpvParam&, ByVal fuWinIni&)

#define SPI_GETFOREGROUNDLOCKTIMEOUT &H2000 '
#define SPI_SETFOREGROUNDLOCKTIMEOUT &H2001 '
#define apiNull 0
```

```
#define apiTrue 1
#define apiFalse 0

Var Shared Timer1 As Object
Timer1.Attach GetDlgItem("Timer1")

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

'=====================================
'=
'=====================================
Declare Sub SetAbsoluteForegroundWindow(ByVal hWnd As Long)
Sub SetAbsoluteForegroundWindow(ByVal hWnd As Long)
    Var nTargetID As Long
    Var nForegroundID As Long
    Var sTime As Long
    Var Ret As Long

    '現在のフォアグラウンドウィンドウを作成したスレッドのIDを取得
    nForegroundID = Api_GetWindowThreadProcessId(Api_GetForegroundWindow(), apiNull)

    '目的のウィンドウを作成したスレッドのIDを取得
    nTargetID = Api_GetWindowThreadProcessId(GethWnd, apiNull)

    '入力処理をアタッチ(入力情報を得る)
    Ret = Api_AttachThreadInput(nTargetID, nForegroundID, apiTrue)

    '現在の設定をsTimeに保存
    Ret = Api_SystemParametersInfo(SPI_GETFOREGROUNDLOCKTIMEOUT, 0, sTime, 0)

    'ウィンドウの切り替え時間を0msに
    Ret = Api_SystemParametersInfo(SPI_SETFOREGROUNDLOCKTIMEOUT, 0, ByVal 0, 0)

    'ウィンドウをフォアグラウンドに
    Ret = Api_SetForegroundWindow(GethWnd)

    '設定を元に戻す
    Ret = Api_SystemParametersInfo(SPI_SETFOREGROUNDLOCKTIMEOUT, 0, ByVal sTime, 0)

    '入力処理をデタッチ(元に戻す)
    Ret = Api_AttachThreadInput(nTargetID, nForegroundID, apiFalse)
End Sub

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    SetAbsoluteForegroundWindow GethWnd
End Sub

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

ウィンドウを最前面に表示（IV）

ウィンドウを最前面に表示します。SetForegroundWindowはWindows95、およびWindows98では正常に機能しますが、Windows2000、WindowsXPにおいてはタスクバーで点滅し（仕様:※注参照）、ウィンドウが隠れてしまいます。（IV参照）
（III）では、AttachThreadInput、Timerを使って強制的に全面表示させています。
GetForegroundWindow ユーザーが操作中のウィンドウを取得
SetForegroundWindow 指定されたウィンドウをフォアグラウンドにしアクティブにする
GetWindowThreadProcessId ウィンドウのプロセスIDとスレッドIDを取得
AttachThreadInput 別スレッドの子ウィンドウにフォーカスをセット
SystemParametersInfo システム全体に関するパラメータを取得・設定

※注
SetForegroundWindow
旧バージョンからの変更により、アプリケーションは、ユーザーが他のウィンドウで作業しているときに強制的にフォアグラウンドウィンドウを設定することはできなくなりました。その代わりに、SetForegroundWindow 関数は、ウィンドウをアクティブにし（SetActiveWindow 関数、FlashWindowEx 関数を呼び出してユーザーに通知します。

```
'====================================
'= ウィンドウを最前面に（IV）
'= Windows95、Windows98でのみ可
'= Windows2000、WindowsXPでは最前面にならずタスクバーで点滅
'=    (SetForegroundWindow3.bas)
'====================================
#include "Windows.bi"

' 指定したウィンドウをフォアグラウンドウィンドウに設定
Declare Function Api_SetForegroundWindow& Lib "user32" Alias "SetForegroundWindow"
(ByVal hWnd&)

' 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&)

' 自分自身のプロセスIDを取得
Declare Function Api_GetCurrentProcessId& Lib "kernel32" Alias "GetCurrentProcessId" ()

Var Shared Timer1 As Object
Timer1.Attach GetDlgItem("Timer1")

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

    Ret = Api_SetProp(GethWnd, "ProcessID", Api_GetCurrentProcessId)

    Timer1.SetInterval 50
    Timer1.Enable -1
End Sub

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

```
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer ()
    Var Ret As Long

    Ret = Api_SetForegroundWindow(GethWnd)
End Sub

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

ウインドウを指定のサイズで復元

ウインドウを指定のサイズで復元します。
GetWindowPlacement ウィンドウの位置・状態を取得
SetWindowPlacement ウィンドウの位置・サイズ・状態を設定
FindWindow 指定した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のハンドルを返す

例では、メモ帳を起動し指定のサイズで復元しています。


```
'=====================================
'= ウインドウを指定のサイズで復元
'=    (SetWindowPlacement.bas)
'=====================================
#include "Windows.bi"

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

Type WINDOWPLACEMENT
    length           As Long
    flags            As Long
    showCmd          As Long
    ptMinPosition    As POINTAPI
    ptMaxPosition    As POINTAPI
    rcNormalPosition As RECT
End Type

' ウィンドウの位置・状態を取得
Declare Function Api_GetWindowPlacement& Lib "user32" Alias "GetWindowPlacement" (ByVal
hWnd&, lpwndpl As WINDOWPLACEMENT)
```

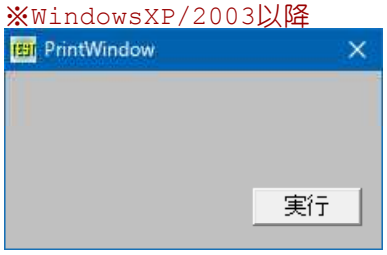
```
' ウィンドウの位置・サイズ・状態を設定
Declare Function Api_SetWindowPlacement& Lib "user32" Alias "SetWindowPlacement" (ByVal
hWnd&, lpwndpl As WINDOWPLACEMENT)

' 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$,
ByVal lpWindowName$)

#define WPF_RESTORETOMAXIMIZED 2          'アイコン化される前に最大表示されていたかどうかにかかわらず、元に戻されるウィンド
#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 SW_SHOWNORMAL 1                   'SW_RESTOREと同じ
#define SW_SHOWMINIMIZED 2                'ウインドウをアクティブ化しアイコン化
#define SW_SHOWMAXIMIZED 3                'ウインドウをアクティブ化し最大表示

Var Shared Radio(2) As Object
Var Shared Button1 As Object
Var Shared Button2 As Object

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

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

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

    'コマンドボタンを無効に設定
    Button1.EnableWindow 0

    'メモ帳を起動
    shell "Notepad.exe",, 5
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var ClassName As String
    Var hWnd As Long
    Var wp As WINDOWPLACEMENT
    Var Ret As Long

    'クラス名でウィンドウハンドルを取得
    ClassName = "Notepad"
    hWnd = Api_FindWindow(ClassName, ByVal 0)

    'ウィンドウハンドルを取得できたときは
    If hWnd <> 0 Then

        'ウィンドウの配置方法に関する情報を定義する構造体のサイズを設定
        wp.length = Len(wp)
```

```
        'ウィンドウの配置方法を取得
        Ret = Api_GetWindowPlacement(hWnd, wp)

        '復元されるウィンドウの最大化を指定
        wp.flags = WPF_RESTORETOMAXIMIZED

        Select Case frmSize
            Case 1
                wp.showCmd = SW_SHOWNORMAL
            Case 2
                wp.showCmd = SW_SHOWMINIMIZED
            Case 3
                wp.showCmd = SW_SHOWMAXIMIZED
        End Select

        'ウィンドウの配置方法を設定
        Ret = Api_SetWindowPlacement(hWnd, wp)
    End If
End Sub

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

ウインドウを指定したデバイスコンテキストにコピー

ウインドウを指定したデバイスコンテキストにコピーします。
PrintWindow 表示されているウィンドウを指定したデバイスコンテキスト(通常はプリンタDC)にコピー
FindWindow ウインドウのハンドル取得

テストでは、フォームにメモ帳のイメージをコピーしています。デバイスコンテキスト(hDC)をButton、Pictureに置き換えるとそれらにイメージをコピーします。
※WindowsXP/2003以降



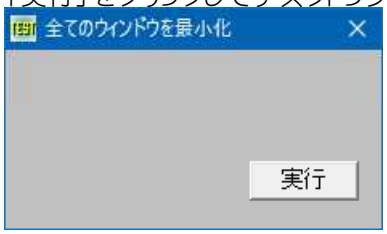

```
'=======================================
'= ウィンドウを指定のDCにコピー(WindowsXP/2003)
'=    (PrintWindow.bas)
'=======================================
#include "Windows.bi"

' 表示されているウィンドウを指定したデバイスコンテキスト(通常はプリンタDC)にコピー
Declare Function Api_PrintWindow& Lib "user32" Alias "PrintWindow" (ByVal hWnd&, ByVal
hdcBlt&, ByVal nFlags&)

' ウインドウのハンドル取得
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 Button1_on edecl ()
Sub Button1_on()
    Var hWnd As Long
    Var hDC As Long
    Var Ret As Long

    hDC = Api_GetDC(GethWnd)                        'フォームのDC取得
    hWnd = Api_FindWindow("Notepad", ByVal 0)       'メモ帳のハンドル取得

    If hWnd = 0 Then                                'メモ帳が見つからない場合は起動
        Shell "Notepad.exe"
        Wait 20
    End If

    hWnd = Api_FindWindow("Notepad", ByVal 0)       'メモ帳のハンドル取得
    Ret = Api_PrintWindow(hWnd, hDC, 0)             'フォームにメモ帳のイメージを描画
    Ret = Api_ReleaseDC(hWnd, hDC)                  'DCの解放
End Sub

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

ウィンドウを全て最小化(Ⅰ)

デスクトップ上のウィンドウを最小化(アイコン化)します。
keybd_event 特殊キーの状態を設定

「実行」をクリックしてデスクトップ上のウィンドウを最小化します。




```
'=====================================
'= ウィンドウを全て最小化
'=    (keybd_event2.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_EXTENDEDKEY &H1          'スキャンコードにプリフィックスバイト0xE0(224)を付加
#define KEYEVENTF_KEYUP &H2                'キーを放す

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Api_keybd_event &H5B, 0, 0, 0                   'Windowsキー
    Api_keybd_event &H4D, 0, 0, 0                   'Dキーを押下
    Api_keybd_event &H5B, 0, KEYEVENTF_KEYUP, 0     'キーを離す
```

```
End Sub

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

ウィンドウを全て最小化（II）

デスクトップ上のウィンドウを全て最小化、それらをもとに戻し(Undo)ます。
FindWindow クラス名、キャプションを与えてウィンドウハンドルを取得
PostMessage 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
SetWindowPos ウィンドウのサイズ、位置、及びZオーダーを設定


```
'=====================================
'= ウィンドウを全て最小化（II）
'=    (PostMessage2.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を作成した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&)

' 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_NOSIZE &H1                          'ウィンドウの現在のサイズを保持する
#define SWP_NOMOVE &H2                          'ウィンドウの現在位置を保持する
#define FLAGS (SWP_NOMOVE Or SWP_NOSIZE)
#define HWND_TOPMOST (-1)                       'ウィンドウを常に最前面に配置
#define HWND_NOTOPMOST (-2)                     'ウィンドウを常に最前面に配置(他のウィンドウが
                                                HWND_TOPMOSTに配置されている場合はその配下)
#define WM_COMMAND &H111                        'メニューが選択された或いはｺﾝﾄﾛｰﾙにイベントが発生した
#define MIN_ALL 419
#define MIN_ALL_UNDO 416
#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

'=====================================
'= フォームを最前面に
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var Ret As Long

    Ret = Api_SetWindowPos(GethWnd, HWND_TOPMOST, 0, 0, GetWidth, GetHeight, FLAGS)
End Sub
```

```
'=====================================
'= 全て最小化
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hWnd As Long
    Var Ret As Long

    hWnd = Api_FindWindow("Shell_TrayWnd", vbNullString)
    Ret = Api_PostMessage(hWnd, WM_COMMAND, MIN_ALL, 0)
End Sub

'=====================================
'= サイズを戻す
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var hWnd As Long
    Var Ret As Long

    hWnd = Api_FindWindow("Shell_TrayWnd", vbNullString)
    Ret = Api_PostMessage(hWnd, WM_COMMAND, MIN_ALL_UNDO, 0)
End Sub

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

ウインドウを操作

例ではForm2をHide・Show・ShowMinNoActive・Restoreさせます。
ShowWindow 指定されたウィンドウの表示状態を設定
SetWindowPos ウィンドウのサイズ、位置、および Z オーダーを設定
左:起動時 SetWindowPosでMainFormを前面に出しています。
中:HIDEでFORM2を消しています。
右:SHOWMINNOACTIVE

FORM2はタスクバーに入っている状態。RESTOREでFORM2を元に戻しています。

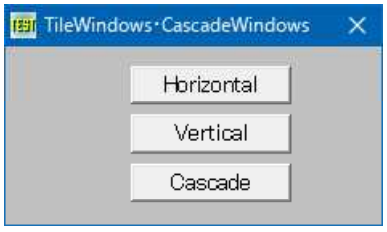

```
'=====================================
'= ウインドウの操作
'=    (ShowWindow2.bas)
'=====================================
#include "Windows.bi"

' 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&)
```

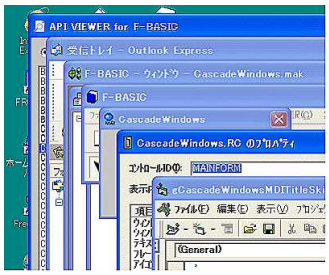
```
#define SW_HIDE 0                                   '指定のウインドウを非表示にし他のウインドウをアクティブ化
#define SW_SHOW 5                                   'ウインドウをアクティブ化し現在の位置とサイズで表示
#define SW_SHOWMINNOACTIVE 7                        'ウインドウをアイコン化する。現在アクティブなウインドウはアクティブなままにする
#define SW_RESTORE 9                                'ウインドウをアクティブ化し表示。ウイ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

' ウィンドウのサイズ、位置、および Z オーダーを設定。(ウィンドウの重なり順のことを「Zオーダー」といいZオーダーのトップに置くと一番手前に表示される)
Declare Function Api_SetWindowPos& Lib "user32" Alias "SetWindowPos" (ByVal hWnd&, ByVal hWndInsertAfter&, ByVal X&, ByVal Y&, ByVal CX&, ByVal CY&, ByVal uFlags&)

#define HWND_TOP 0                                  'ウインドウをZオーダーの一番上に配置する
#define HWND_BOTTOM 1                               'ウインドウをウインドウリストの一番下に配置する
#define HWND_TOPMOST -1                             'ウインドウをウインドウリストの一番上に配置する
#define HWND_NOTOPMOST -2                           'ウインドウをウインドウリストの一番上に配置する

#define SWP_SHOWWINDOW &H40                         'ウインドウを表示する
#define SWP_NOSIZE &H1                              'ウインドウの現在のサイズを保持する
#define SWP_NOMOVE &H2                              'ウインドウの現在位置を保持する

Var Shared Form2 As Object

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

    If Not (CheckObject(Form2)) Then
        Form2.CreateWindow "Form2", -1
    End If
    Ret = Api_SetWindowPos(GethWnd, HWND_TOPMOST, 0, 0, 0,0, SWP_SHOWWINDOW Or SWP_NOMOVE Or SWP_NOSIZE)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long
    Ret = Api_ShowWindow(Form2.GethWnd, SW_HIDE)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Ret As Long
    Ret = Api_ShowWindow(Form2.GethWnd, SW_SHOW)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button3_on edecl ()
Sub Button3_on()
    Var Ret As Long
    Ret = Api_ShowWindow(Form2.GethWnd, SW_SHOWMINNOACTIVE)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button4_on edecl ()
Sub Button4_on()
```

```
    Var Ret As Long
    Ret = Api_ShowWindow(Form2.GethWnd, SW_RESTORE)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button5_on edecl ()
Sub Button5_on()
    End
End Sub

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

ウィンドウを縦横に並べて、または重ねて表示

TileWindows 指定されたウィンドウまたは指定された親ウィンドウの子ウィンドウを並べて表示
CascadeWindows 指定された親ウィンドウの指定された子ウィンドウを重ねて表示





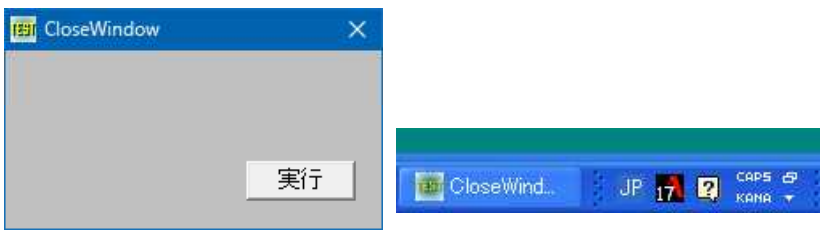
図は「Cascade」の選択状態です。

```
'=====================================
'= ウィンドウを縦横に並べて、または重ねて表示
'=     (TileWindows.bas)
'=====================================
#include "Windows.bi"

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

#define MDITILE_VERTICAL &H0                'MDIクライアントウィンドウの高さ一杯に左右に並べて表示
#define MDITILE_HORIZONTAL &H1              'MDIクライアントウィンドウの横幅一杯に上下に並べて表示
#define MDITILE_SKIPDISABLED &H2            '使用禁止状態のMDI子ウィンドウを除外
#define MDITILE_ZORDER &H4                  'ウィンドウはZオーダーに従って整列

' 指定されたウィンドウまたは指定された親ウィンドウの子ウィンドウを並べて表示する
Declare Function Api_TileWindows& Lib "user32" Alias "TileWindows" (ByVal hwndParent&,
ByVal wHow&, lpRect As RECT, ByVal cKids&, lpKids&)

' 指定された親ウィンドウの指定された子ウィンドウを重ねて表示
Declare Function Api_CascadeWindows& Lib "user32" Alias "CascadeWindows" (ByVal
hwndParent&, ByVal wHow&, lpRect As RECT, ByVal cKids&, lpkids&)

Var Shared rc As RECT
```

```
'====================================
'= ウィンドウを整列する長方形を指定
'====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()

    rc.left = 0
    rc.top = 0
    rc.right = 600
    rc.bottom = 480
End Sub

'====================================
'= デスクトップ上のウィンドウを横に整列
'====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Ret As Long

    Ret = Api_TileWindows(0, MDITILE_HORIZONTAL, rc, 0, ByVal 0)
End Sub

'====================================
'= デスクトップ上のウィンドウを縦に整列
'====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Ret As Long

    Ret = Api_TileWindows(0, MDITILE_VERTICAL, rc, 0, ByVal 0)
End Sub

'====================================
'= デスクトップ上のウィンドウを重ねて表示
'====================================
Declare Sub Button3_on edecl ()
Sub Button3_on()
    Var Ret As Long

    Ret = Api_CascadeWindows(0, MDITILE_SKIPDISABLED, rc, 0, ByVal 0)
End Sub

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

ウィンドウを最小化

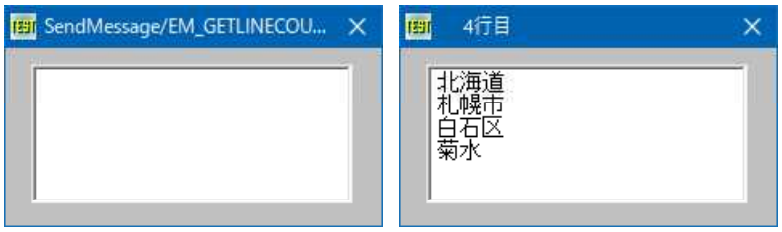
ウィンドウを最小化します。単語から察すると終了かと思ってしまいます。
CloseWindow ウィンドウを最小化



```
'=====================================
'= ウィンドウを最小化する
'=     (CloseWindow.bas)
'=====================================
#include "Windows.bi"

' ウィンドウを最小化する
Declare Function Api_CloseWindow& Lib "user32" Alias "CloseWindow" (ByVal hWnd&)

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

    Ret = Api_CloseWindow(GethWnd)
End Sub

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

エディットボックス内のカーソル行を取得

SendMessage ウィンドウにメッセージを送信
EM_GETLINECOUNT(&HBA) MLE(複数行編集)内の行数を取得



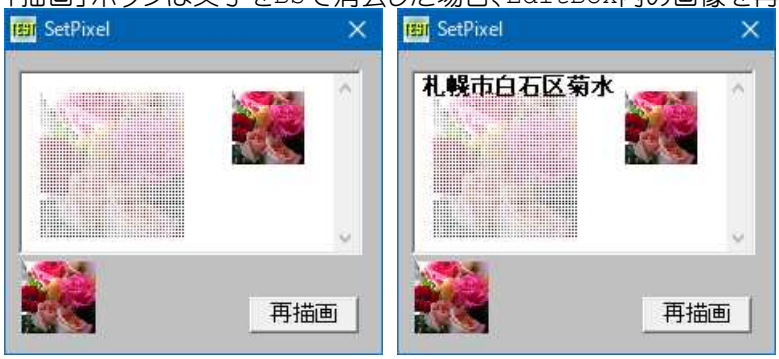

```
'=====================================
'= エディットボックス内のカーソル行を取得
'=    (SendMessage10.bas)
'=====================================
#include "Windows.bi"

' 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 EM_GETLINECOUNT &HBA                'MLE(複数行編集)内の行数を取得
Var Shared Edit1 As Object

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

'=====================================
'=
'=====================================
Declare Sub Edit1_Change edecl ()
Sub Edit1_Change()
    Var lineCount As Long

    lineCount = Api_SendMessage(Edit1.GethWnd, EM_GETLINECOUNT, 0, ByVal 0)
    SetWindowText Format$(lineCount, "##,###行目")
End Sub
```

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

エディットボックスに画像転写

エディットボックスに画像を転写してみます。
GetDC デバイスコンテキストを取得
SetPixel 指定した座標に点を配置
GetPixel 指定された座標のRGB値を取得
ReleaseDC デバイスコンテキストを解放

Picture1にflower.bmpを描画し、EditBoxに2倍のピクセル間隔で転写しています。
「描画」ボタンは文字をBSで消去した場合、EditBox内の画像を再描画するものです。



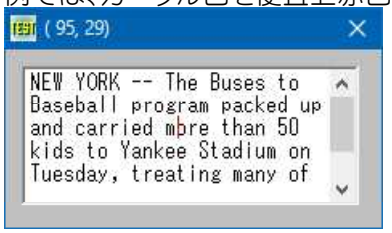

```
'=====================================
'= エディットボックスに画像転写
'=    (SetPixel.bas)
'=====================================
#include "Windows.bi"

' 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&)

' 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&)

' 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 Button1 As Object
Var Shared Picture1 As Object
Var Shared Bitmap As Object

BitmapObject Bitmap
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Picture1.Attach GetDlgItem("Picture1")

Var Shared EhDC As Long
Var Shared PhDC As Long

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

```
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var i As Long
    Var j As Long
    Var Col As Long
    Var Ret As Long

    For i = 1 To Picture1.GetWidth - 1
        For j = 1 To Picture1.GetHeight - 1
            Col = Api_GetPixel(PhDC, i, j)
            Ret = Api_SetPixel(EhDC, 10 + i * 2, 10 + j * 2, Col)
            Ret = Api_SetPixel(EhDC, 130 + i, 10 + j, Col)
        Next
    Next
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Bitmap.LoadFile "flower.bmp"
    Picture1.StretchBitmap Bitmap, 0, 0, 46, 46
    Bitmap.DeleteObject

    EhDC = Api_GetDC(Edit1.GethWnd)
    PhDC = Api_GetDC(Picture1.GethWnd)

    Button1_on

    Edit1.SetFocus
End Sub

'=====================================
'=
'=====================================
Declare Sub Edit1_Change edecl ()
Sub Edit1_Change()
    Button1_on
End Sub

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

    Ret = Api_ReleaseDC(Edit1.GethWnd, EhDC)
    Ret = Api_ReleaseDC(Picture1.GethWnd, PhDC)
    End
End Sub

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

エディットボックスのカーソル座標を取得

EditBoxの文字座標を取得します。
SendMessage ウィンドウにメッセージを送信
EM_POSFROMCHAR(&HD6) 指定の文字インデックス座標を取得

例では、カーソル色を便宜上赤色で表しています。



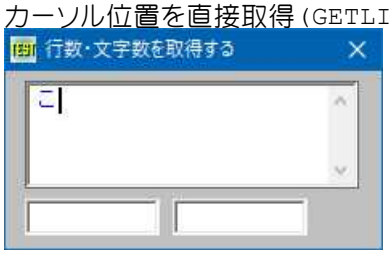

```
'=======================================
'= エディットボックスのカーソル座標を取得
'=    (EM_POSFROMCHAR.bas)
'=======================================
#include "Windows.bi"

Type POINTAPI
    x   As Long
    y   As Long
End Type

' 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_POSFROMCHAR &HD6                  '指定の文字インデックス座標を取得

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()
    Var txt As String
    txt = "NEW YORK -- The Buses to Baseball program packed up and carried more than 50 kids to Yankee Stadium on Tuesday, treating many of them to their first Major League Baseball game."
    Edit1.SetWindowtext txt

    Timer1.SetInterval 50
    Timer1.Enable -1
End Sub

'=======================================
'=
'=======================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var PosFromChar As Long
    Var pa As POINTAPI
    Var CursorPos As String
    Var Ret As Long

    'カーソル位置の文字番目を指定
    PosFromChar = Edit1.GetSelTextStart

    '指定文字番目の座標を取得
    Ret = Api_SendMessage(Edit1.GethWnd, EM_POSFROMCHAR, PosFromChar, ByVal CLng(0))

    '指定文字番目が取得範囲内
    If PosFromChar < Len(Edit1.GetWindowText) Then

        'カーソル座標を算出
        pa.x = Ret And &H7FFF
```

```
        pa.y = Ret ¥ (2 ^ 16)

        CursorPos = "(" & Str$(pa.x) & "," & Str$(pa.y) & ")"

        '座標を表示
        SetWindowText CursorPos

    'カーソル位置が取得範囲外
    Else

        'エラーを表示
        SetWindowText "取得範囲外！"
    End If
End Sub

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

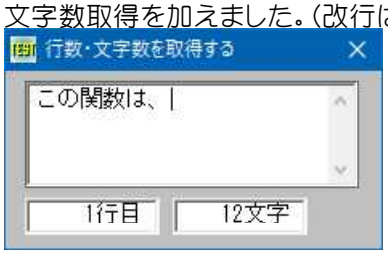
エディットボックスの行数･文字数を取得

マルチラインエディットボックスの行数を取得します。
SendMessage ウィンドウにメッセージを送信
EM_GETLINECOUNT(&HBA) マルチラインエディットボックスの行数を取得
EM_LINEFROMCHAR(&HC9) 文字インデックスから行番号を取得する

カーソル位置を直接取得(GETLINECOUNT)、インデックスから行番号を取得(LINEFROMCHAR)



文字数取得を加えました。(改行は半角換算2文字加算されます)



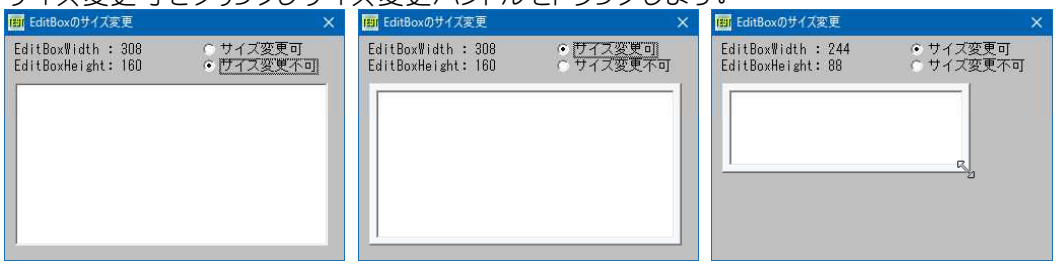

```
'=====================================
'= 行数･文字数を取得する
'=    (GetLineCount.bas)
'=====================================
#include "Windows.bi"

' 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_GETLINECOUNT &HBA                'MLE内の行数を取得する
#define EM_LINEINDEX &HBB                   'MLEの行の文字インデックスを取得する
#define EM_LINELENGTH &HC1                  'MLE内の行の長さを取得する

Var Shared Edit1 As Object
Var Shared Text1 As Object
```

```
Var Shared Text2 As Object
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Text2.Attach GetDlgItem("Text2") : Text2.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Edit1_Change edecl ()
Sub Edit1_Change()
    Var lineCount As Long
    Var ChrsUpToLast As Long
    Var DocSize As Long

    lineCount = Api_SendMessage(Edit1.Gethwnd, EM_GETLINECOUNT, 0, ByVal 0)
    Text1.SetWindowText Format$(lineCount, "##,###行目")

    ChrsUpToLast = Api_SendMessage(Edit1.GethWnd, EM_LINEINDEX, lineCount - 1, ByVal 0)
    If ChrsUpToLast = 0 Then
        DocSize = Api_SendMessage(Edit1.GethWnd, EM_LINELENGTH, ChrsUpToLast, ByVal 0)
    Else If ChrsUpToLast < 65000 Then
        DocSize = Api_SendMessage(Edit1.GethWnd, EM_LINELENGTH, ChrsUpToLast, ByVal 0) +
ChrsUpToLast
    End If
    Text2.SetWindowText Format$(DocSize, "##,###文字")
End Sub

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

エディットボックスのサイズ変更

エディとボックスのサイズを変更してみます。
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
SetWindowPos ウィンドウのサイズ、位置、および Z オーダーを設定

サイズ変更可をクリックしサイズ変更ハンドルをドラッグします。




```
'=====================================
'= エディットボックスのサイズ変更
'=    (ResizeEditBox.bas))
'=====================================
#include "Windows.bi"

#define SWP_DRAWFRAME &H20                  '再描画のときウインドウを囲む枠も描画
#define SWP_NOMOVE &H2                      'ウインドウの現在位置を保持する
#define SWP_NOSIZE &H1                      'ウインドウの現在のサイズを保持する
#define SWP_NOZORDER &H4                    'ウインドウリスト内での現在位置を保持する
#define WS_THICKFRAME &H40000               'サイズ変更境界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
```

```
' 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&)

' ウィンドウのサイズ、位置、および Z オーダーを設定。(ウィンドウの重なり順のことを「Zオーダー」といいZオーダーのトップに置くと一番手前に表示される)
Declare Function Api_SetWindowPos& Lib "user32" Alias "SetWindowPos" (ByVal hWnd&, ByVal
hWndInsertAfter&, ByVal X&, ByVal Y&, ByVal CX&, ByVal CY&, ByVal uFlags&)

Var Shared Edit1 As Object
Var Shared Text1 As Object
Var Shared Text2 As Object
Var Shared Radio1 As Object
Var Shared Radio2 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 12
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Text2.Attach GetDlgItem("Text2") : Text2.SetFontSize 14
Radio1.Attach GetDlgItem("Radio1") : Radio1.SetFontSize 14
Radio2.Attach GetDlgItem("Radio2") : Radio2.SetFontSize 14

Var Shared iniStyle As Long

'=====================================
'=
'=====================================
Declare Sub initStyle_DSP edecl ()
Sub initStyle_DSP()
    If iniStyle Then
        Ret = Api_SetWindowLong(Edit1.GethWnd, GWL_STYLE, iniStyle)
        Ret = Api_SetWindowPos(Edit1.GethWnd, GethWnd, 0, 0, 0, 0, SWP_NOZORDER Or
SWP_NOSIZE Or SWP_NOMOVE Or SWP_DRAWFRAME)
    End If
End Sub

'=====================================
'=
'=====================================
Declare Sub Size_DSP edecl ()
Sub Size_DSP()
    Text1.SetWindowText "EditBoxWidth :" & Str$(Edit1.GetWidth)
    Text2.SetWindowText "EditBoxHeight:" & Str$(Edit1.GetHeight)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    iniStyle = Api_GetWindowLong(Edit1.GethWnd, GWL_STYLE)

    initStyle_DSP
    Size_DSP
End Sub

'=====================================
'=
'=====================================
Declare Sub Radio1_on edecl ()
Sub Radio1_on()
    Var newStyle As Long

    newStyle = Api_GetWindowLong(Edit1.GethWnd, GWL_STYLE)
    newStyle = newStyle Or WS_THICKFRAME
```

```
     If newStyle Then
        Ret = Api_SetWindowLong(Edit1.GethWnd, GWL_STYLE, newStyle)
        Ret = Api_SetWindowPos(Edit1.GethWnd, GethWnd, 0, 0, 0, 0, SWP_NOZORDER Or
SWP_NOSIZE Or SWP_NOMOVE Or SWP_DRAWFRAME)
    End If

    Size_DSP
End Sub

'=====================================
'=
'=====================================
Declare Sub Radio2_on edecl ()
Sub Radio2_on()
    initStyle_DSP

    Size_DSP
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_MouseMove edecl ()
Sub MainForm_MouseMove()
    Size_DSP
End Sub

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

エディットボックスの属性を変更

GetWindowLong 指定されたウィンドウに関しての情報を取得
SetWindowLong 指定されたウィンドウの属性を変更
SendMessage ウィンドウにメッセージを送信

例では、エディットボックスの入力時に大文字での入力・小文字での入力・数字のみの入力を切り替えます。

```
'=====================================
'= エディットボックスの属性を変更
'=    (SetWindowLong2.bas)
'=====================================
#include "Windows.bi"

' 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&)

' 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 GWL_STYLE -16                          'アプリケーションのインスタンスハンドル
#define ES_LOWERCASE &H10                      '入力文字を小文字にする
#define ES_NUMBER &H2000                       '数値入力専用にする
#define ES_UPPERCASE 8                         '入力文字を大文字にする
#define EM_SCROLLCARET &HB7                    'キャレットをスクロールさせて表示す
#define EM_SETREADONLY &HCF                    'エディットコントロールの読み取り専用スタイルを設定する
#define EM_SETSEL &HB1                         'エディットコントロール内部のテキストを選択する
#define EM_UNDO &HC7                           'エディットコントロール内での直前の操作を取り消す

Var Shared Edit1 As Object
Var Shared Radio(3) As Object
Var Shared Check1 As Object
Var Shared Group(1) As Object
Var Shared Button(1) As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
For i = 0 To 3
    If i < 2 Then
        Group(i).Attach GetDlgItem("Group" & Trim$(Str$(i + 1))) : Group(i).SetFontSize
14
        Button(i).Attach GetDlgItem("Button" & Trim$(Str$(i + 1))) :
Button(i).SetFontSize 14
    End If
    Radio(i).Attach GetDlgItem("Radio" & Trim$(Str$(i + 1))) : Radio(i).SetFontSize 14
Next
Check1.Attach GetDlgItem("Check1") : CHeck1.SetFontSize 14

Var Shared defstyle As Long

'====================================
'=
'====================================
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()
   defstyle = Api_GetWindowLong(Edit1.GethWnd, GWL_STYLE)
   ShowWindow -1
End Sub

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

    Ret = Api_SendMessage(Edit1.GethWnd, EM_UNDO, 0, ByVal 0)

    Edit1.SetFocus
End Sub

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

```
    Ret = Api_SendMessage(Edit1.GethWnd, EM_SCROLLCARET, 0, ByVal 0)
    Edit1.SetFocus
End Sub

'=====================================
'=
'=====================================
Declare Sub RadioSel()
Sub RadioSel()
    Var Ret As Long

    Ret = Api_SetWindowLong(Edit1.GethWnd, GWL_STYLE, defstyle)

    Select Case index
        Case 0                              '通常

        Case 1                              '数字のみ(ES_NUMBER)
            Ret = Api_SetWindowLong(Edit1.GethWnd, GWL_STYLE, defstyle Or ES_NUMBER)

        Case 2                              '大文字(ES_UPPERCASE)
            Ret = Api_SetWindowLong(Edit1.GethWnd, GWL_STYLE, defstyle Or ES_UPPERCASE)

        Case 3                              '小文字(ES_LOWERCASE)
            Ret = Api_SetWindowLong(Edit1.GethWnd, GWL_STYLE, defstyle Or ES_LOWERCASE)
    End Select

    Edit1.SetFocus
End Sub

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

    state = Check1.GetCheck
    Ret = Api_SendMessage(Edit1.GethWnd, EM_SETREADONLY, state, ByVal 0)

    Edit1.SetFocus
End Sub

'=====================================
'=
'=====================================
Declare Sub Radio1_on edecl ()
Sub Radio1_on()
    RadioSel
End Sub

'=====================================
'=
'=====================================
Declare Sub Radio2_on edecl ()
Sub Radio2_on()
    RadioSel
End Sub

'=====================================
'=
'=====================================
Declare Sub Radio3_on edecl ()
Sub Radio3_on()
    RadioSel
End Sub

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

```
Declare Sub Radio4_on edecl ()
Sub Radio4_on()
    RadioSel
End Sub

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

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

エディットボックスの入力可能最大バイト数を取得

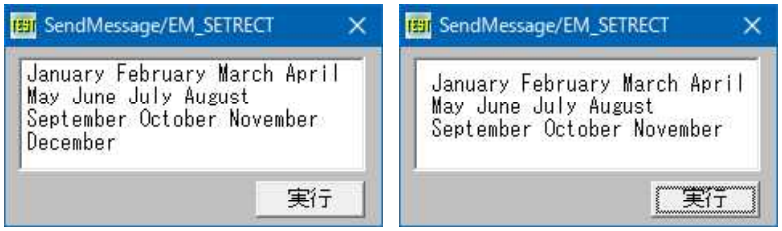
SendMessage ウィンドウにメッセージを送信
EM_GETLIMITTEXT(&HD5) エディットボックスの入力可能最大バイト数を取得
EM_LIMITTEXT(&HC5) エディットコントロール内のテキストの文字数を制限


```
'=====================================
'= エディットボックスの入力可能最大バイト数を取得
'=    (EM_GETLIMITTEXT.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 EM_GETLIMITTEXT &HD5                'エディットボックスの入力可能最大バイト数を取得
#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 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 LimitText As Long
    Var GetLimitText As Long
    Var Ret As Long
```

```
    If Edit1.GetWindowText <> "" Then
        LimitText = Val(Edit1.GetWindowText)
        Ret = Api_SendMessage(Edit1.GethWnd, EM_LIMITTEXT, LimitText, ByVal CLng(0))
    End If

    Edit1.SetWindowText ""
    Edit1.SetWindowText String$(LimitText, "@")

    'テキストボックスに入力可能な最大バイト数を取得
    GetLimitText = Api_SendMessage(Edit1.GethWnd, EM_GETLIMITTEXT, 0, ByVal CLng(0))

    '結果を表示
    Text1.SetWindowText "最大" & Str$(GetLimitText) & "バイト"
End Sub


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

エディットボックスのフォーマット矩形を設定

SendMessage ウィンドウにメッセージを送信
EM_GETRECT(&HB2) エディットコントロール長方形の座標を取得する
EM_SETRECT(&HB3) MLEの書式化長方形を設定する




```
'=======================================
'= エディットボックスのフォーマット矩形を設定
'=    (SendMessage11.bas)
'=======================================
#include "Windows.bi"

Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type
#define EM_GETRECT &HB2                    'エディットコントロール長方形の座標を取得する
#define EM_SETRECT &HB3                    'MLEの書式化長方形を設定する

' 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)

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 txt As String
```

```
'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    txt = "January February March April May June July August September October November
December"
    Edit1.SetWindowText txt
End Sub

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

    'フォーマット矩形を取得
    Ret = Api_SendMessage(Edit1.GethWnd, EM_GETRECT, 0, rc)

    '新しいフォーマット矩形を指定
    rc.Top = rc.Top + 5
    rc.Left = rc.Left + 5

    '新しいフォーマット矩形を設定
    Ret = Api_SendMessage(Edit1.GethWnd, EM_SETRECT, 0, rc)

    'テキストボックスを更新
    Edit1.SetWindowtext txt
End Sub

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

エディットボックスの編集有無を判定

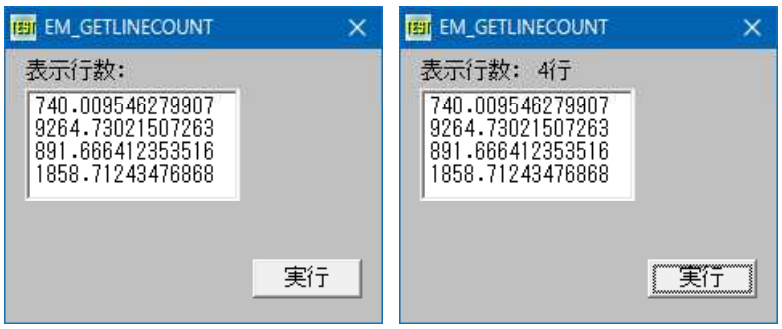
エディットボックスの内容が編集されたかどうかを判定します。
SendMessage ウィンドウにメッセージを送信
EM_GETMODIFY(&HB8) エディットコントロールの内容が変更されたかどうかをチェック


```
'=====================================
'= エディットボックスの編集有無を判定
'=    (EM_GETMODIFY.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_GETMODIFY &HB8                    'エディットコントロールの内容が変更されたかどうかをチェック

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 Ret As Long

    'エディットボックスの編集有無を判定
    Ret = Api_SendMessage(Edit1.GethWnd, EM_GETMODIFY, 0, ByVal CLng(0))

    '判定結果表示
    If Ret = False Then
        Text1.SetWindowText "編集されていません。"
    Else
        Text1.SetWindowText "編集されました。"
    End If
End Sub

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

エディットボックス(複数行入力)の表示行数取得

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
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
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
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
GetTextMetrics フォントに関する情報を取得
ReleaseDC デバイスコンテキストを解放
EM_GETLINECOUNT(&HBA) MLE(複数行編集)内の行数を取得




```
'=====================================
'= エディットボックス(複数行入力)の表示行数取得
'=    (EM_GETLINECOUNT.bas)
'=====================================
#include "Windows.bi"
```

```
Type TEXTMETRIC
    tmHeight                As Long
    tmAscent                As Long
    tmDescent               As Long
    tmInternalLeading       As Long
    tmExternalLeading       As Long
    tmAveCharWidth          As Long
    tmMaxCharWidth          As Long
    tmWeight                As Long
    tmOverhang              As Long
    tmDigitizedAspectX      As Long
    tmDigitizedAspectY      As Long
    tmFirstChar             As Byte
    tmLastChar              As Byte
    tmDefaultChar           As Byte
    tmBreakChar             As Byte
    tmItalic                As Byte
    tmUnderlined            As Byte
    tmStruckOut             As Byte
    tmPitchAndFamily        As Byte
    tmCharSet               As Byte
End Type

Type RECT
    Left        As Long
    Top         As Long
    Right       As Long
    Bottom      As Long
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&)

' ウィンドウのサイズ、位置、および Z オーダーを設定。(ウィンドウの重なり順のことを「Zオーダー」といいZオーダーのトップに置くと一番手前に表示される)
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
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&)

' 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)

' 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 GWL_STYLE -16                       'アプリケーションのインスタンスハンドル
#define WS_THICKFRAME &H40000               'サイズ変更境界を持つウィンドウを作成する
#define SWP_DRAWFRAME &H20                  '再描画のときウインドウを囲む枠も描画
#define SWP_FRAMECHANGED &H20               'ウインドウのサイズ変更中でなくてもWM_NCCALCSIZEを
                                            送る
```

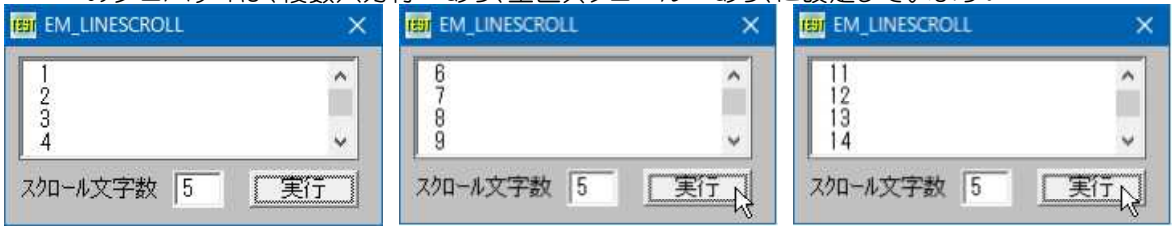
```
#define SWP_HIDEWINDOW &H80                 'ウインドウを隠す
#define SWP_NOACTIVATE &H10                 'ウインドウをアクティブにしない
#define SWP_NOCOPYBITS &H100                'クライアント領域の内容をクリアする
#define SWP_NOMOVE &H2                      'ウインドウの現在位置を保持する
#define SWP_NOOWNERZORDER &H200             'オーナーウインドウのZオーダーは変えない
#define SWP_NOREDRAW &H8                    'ウインドウを自動的に再描画しない
#define SWP_NOREPOSITION &H200              'SWP_NOOWNERZORDERと同じ
#define SWP_NOSIZE &H1                      'ウインドウの現在のサイズを保持する
#define SWP_NOZORDER &H4                    'ウインドウリスト内での現在位置を保持する
#define SWP_SHOWWINDOW &H40                 'ウインドウを表示する
#define SWP_FLAGS (SWP_NOZORDER Or SWP_NOSIZE Or SWP_NOMOVE Or SWP_DRAWFRAME)
#define EM_GETRECT &HB2                     'エディットコントロール長方形の座標を取得する
#define WM_GETFONT &H31                     'テキストボックス・エディットボックス等が現在使っているフォントのハンドル
#define EM_GETLINECOUNT &HBA                'MLE(複数行編集)内の行数を取得する

Var Shared Text1 As Object
Var Shared Text2 As Object
var Shared Edit1 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Text2.Attach GetDlgItem("Text2") : Text2.SetFontSize 14
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var newStyle As Long
    Var txt As String
    Var Ret As Long

    For i = 1 To 20
        txt = txt & Trim$(Str$(Rnd(1) * 10000)) & Chr$(13, 10)
    Next

    Edit1.SetWindowtext txt

    'エディットボックスのサイズを変更可能に
    newStyle = Api_GetWindowLong(Edit1.GethWnd, GWL_STYLE)
    newStyle = newStyle Or WS_THICKFRAME

    Ret = Api_SetWindowLong(Edit1.GethWnd, GWL_STYLE, newStyle)
    Ret = Api_SetWindowPos(Edit1.GethWnd, GethWnd, 0, 0, 0, 0, SWP_FLAGS)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var rct As RECT
    Var TextBoxHeihgt As Long
    Var hTextBoxDC As Long
    Var hFont As Long
    Var tm As TEXTMETRIC
    Var TextboxLineCount As Long
    Var VisibleLineCount As Long
    Var Ret As Long

    'フォーマット矩形を取得
    Ret = Api_SendMessage(Edit1.GethWnd, EM_GETRECT, 0, rct)

    '描画高を算出
    TextBoxHeihgt = rct.Bottom - rct.Top
```

```
    'デバイスコンテキストのハンドルを取得
    hTextBoxDC = Api_GetDC(Edit1.GethWnd)

    '使用しているフォントのハンドルを取得
    hFont = Api_SendMessage(Edit1.GethWnd, WM_GETFONT, 0, ByVal CLng(0))

    'システムフォント以外のとき
    If hFont <> 0 Then

        'フォントオブジェクトを選択
        Ret = Api_SelectObject(hTextBoxDC, hFont)
    End If

    'テキストボックスのテキストメトリック構造体を取得
    Ret = Api_GetTextMetrics(hTextBoxDC, tm)

    'デバイスコンテキストのハンドルを開放
    Ret = Api_ReleaseDC(Edit1.gethWnd, hTextBoxDC)

    'テキストボックス全体の行数を取得
    TextboxLineCount = Api_SendMessage(Edit1.GethWnd, EM_GETLINECOUNT, 0, ByVal CLng(0))

    '取得した情報から表示されている行数を算出
    VisibleLineCount = TextBoxHeihgt ¥ tm.tmHeight

    '算出された行数よりも全体の行数が少ないとき
    If VisibleLineCount > TextboxLineCount Then

        'テキストボックス全体の行数が表示されている行
        VisibleLineCount = TextboxLineCount
    End If

    '結果を表示
    Text2.SetWindowText Str$(VisibleLineCount) & "行"
End Sub

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

エディットボックスを垂直スクロールさせる

SendMessage ウィンドウにメッセージを送信
EM_LINESCROLL(&HB6) MLE内のテキストをスクロールさせる

スクロール文字数を入力し「Enter」押下でEditBoxの文字列およびスクロールバーを戻しています。
Edit1のプロパティは、複数入力行→あり、垂直スクロール→あり、に設定しています。




```
'=====================================
'= エディットボックスを垂直スクロールさせる
'=    (EM_LINESCROLL2.bas)
'=====================================
#include "Windows.bi"
```

```
' スクロールバーのスライダ位置を設定
Declare Function Api_SetScrollPos& Lib "user32" Alias "SetScrollPos" (ByVal hWnd&, ByVal
nBar&, ByVal nPos&, ByVal bRedraw&)

' 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 EM_LINESCROLL &HB6                    'MLE内のテキストをスクロールさせる
#define CrLf Chr$(13, 10)

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()
    Var i As Long
    Var txt As String

    For i = 1 To 100
        txt = txt & Str$(i) & CrLf
    Next
    Edit1.SetWindowText txt
End Sub

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

    '垂直スクロールする文字の数を指定
    ScrlNum = Val(Edit2.GetWindowtext)

    '垂直スクロール実行
    Ret = Api_SendMessage(Edit1.GethWnd, EM_LINESCROLL, 0, ByVal ScrlNum)
End Sub

'=====================================
'= 数値 +「Enter」で決定
'=====================================
Declare Sub Edit2_Change edecl ()
Sub Edit2_Change()
    ED$ = Edit2.GetWindowText
    EPos% = InStr(ED$, CrLf)
    If EPos% <> 0 Then
        ED$ = Mid$(ED$, 1, EPos% - 1) & Mid$(ED$, EPos% + 2)
        Edit2.SetWindowText ED$

        Ret& = Api_SetScrollPos(Edit1.GethWnd, SB_HORZ, 0, 1)
        Edit1.SetSelText 0, 0, 1
        Edit1.SetFocus
    End If
End Sub

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

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

エディットボックスを水平スクロールさせる

SendMessage ウィンドウにメッセージを送信
SB_HORZ(0) 標準スクロールバーの水平
EM_LINESCROLL(&HB6) MLE内のテキストをスクロールさせる

スクロール文字数を入力し「Enter」押下でEditBoxの文字列およびスクロールバーを戻しています。
Edit1のプロパティは、複数入力行→あり、垂直オートスクロール→あり、に設定しています。

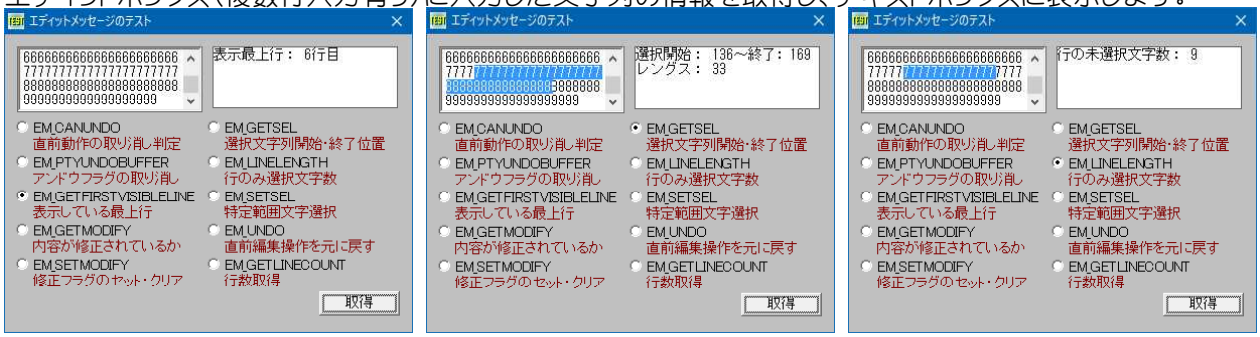

```
'=====================================
'= エディットボックスを水平スクロールさせる
'=    (EM_LINESCROLL.bas)
'=====================================
#include "Windows.bi"

' スクロールバーのスライダ位置を設定
Declare Function Api_SetScrollPos& Lib "user32" Alias "SetScrollPos" (ByVal hWnd&, ByVal
nBar&, ByVal nPos&, ByVal bRedraw&)

' 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 SB_HORZ 0                          '標準スクロールバーの水平
#define EM_LINESCROLL &HB6                 'MLE内のテキストをスクロールさせる
#define CrLf Chr$(13, 10)

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 Button1_on edecl ()
Sub Button1_on()
    Var ScrlNum As Long
    Var Ret As Long

    '水平スクロールする文字の数を指定
    ScrlNum = Val(Edit2.GetWindowtext)

    '水平スクロール実行
    Ret = Api_SendMessage(Edit1.GethWnd, EM_LINESCROLL, ScrlNum, ByVal CLng(0))
End Sub
```

```
'=====================================
'= 数値 +「Enter」で決定
'=====================================
Declare Sub Edit2_Change edecl ()
Sub Edit2_Change()
    ED$ = Edit2.GetWindowText
    EPos% = InStr(ED$, CrLf)
    If EPos% <> 0 Then
        ED$ = Mid$(ED$, 1, EPos% - 1) & Mid$(ED$, EPos% + 2)
        Edit2.SetWindowText ED$

        Ret& = Api_SetScrollPos(Edit1.GethWnd, SB_HORZ, 0, 1)
        Edit1.SetSelText 0, 0, 1
        Edit1.SetFocus
    End If
End Sub

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

エディットメッセージの確認

エディットボックスにメッセージ（VBではテキストメッセージ）を送り結果をテキストボックスに表示します。
SendMessage ウィンドウにメッセージを送信

エディットボックス（複数行入力有り）に入力した文字列の情報を取得し、テキストボックスに表示します。




```
'=====================================
'= エディットメッセージの確認
'=     (EditBoxMulti.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&)

#define EM_CANUNDO &HC6                    'エディットコントロールの操作を取り消せるかどうかを判断する

#define EM_EMPTYUNDOBUFFER &HCD            'エディットコントロールのアンドゥフラグをリセット（クリア）
#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
#define EM_GETFIRSTVISIBLELINE &HCE        'エディットコントロール内の最初の行のインデックスを取得
#define EM_GETHANDLE &HBD                  'MLE用メモリのハンドルを取得するEM_GETLINEMLEから1行取得する

#define EM_GETLINE &HC4                    '指定行の文字列を取得
#define EM_GETLINECOUNT &HBA               'MLE内の行数を取得する
#define EM_GETMARGINS &HD4                 '左右マージンの取得
#define EM_GETMODIFY &HB8                  'エディットコントロールの内容が変更されたかどうかをチェックする
```

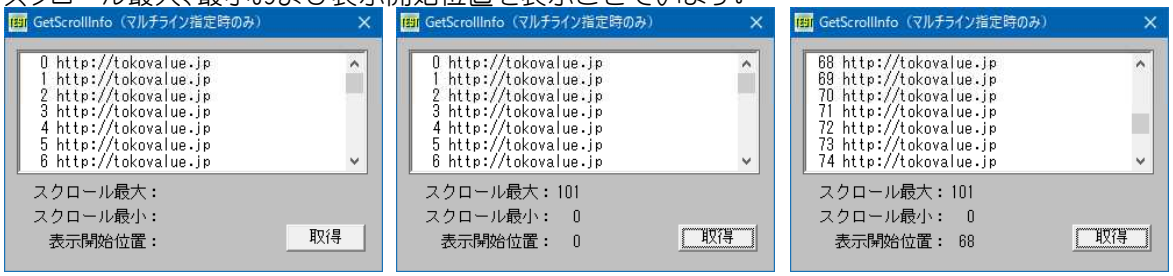
```
#define EM_GETPASSWORDCHAR &HD2             'エディットコントロールのパスワード文字を取得する
#define EM_GETRECT &HB2                     'エディットコントロール長方形の座標を取得する
#define EM_GETSEL &HB0                      'エディットコントロールの現在の選択項目の位置を取得
#define EM_GETTHUMB &HBE                    'スクロールバーの位置を取得
#define EM_GETWORDBREAKPROC &HD1            'エディットコントロールのワードラップ関数を取得する
#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
#define EM_LINEFROMCHAR &HC9                '文字インデックスから行番号を取得する
#define EM_LINEINDEX &HBB                   'MLEの行の文字インデックスを取得する
#define EM_LINELENGTH &HC1                  'MLE内の行の長さを取得する
#define EM_LINESCROLL &HB6                  'MLE内のテキストをスクロールさせる
#define EM_REPLACESEL &HC2                  'エディットコントロール内の現在の選択項目を置き換える
#define EM_SCROLL &HB                       'MLEを垂直にスクロールさせる
#define EM_SCROLLCARET &HB7                 'キャレットをスクロールさせて表示する
#define EM_SETHANDLE &HBC                   'MLEのメモリハンドルを設定する
#define EM_SETMARGINS &HD3                  '左右マージンの設定
#define EM_SETMODIFY &HB9                   'エディットコントロールの変更フラグをセットまたはクリア
#define EM_SETPASSWORDCHAR &HCC             'エディットコントロールのパスワード文字を設定または削除
#define EM_SETREADONLY &HCF                 'エディットコントロールの読み取り専用スタイルを設定
#define EM_SETRECT &HB3                     'MLEの書式化長方形を設定する
#define EM_SETRECTNP &HB4                   'MLEの書式化長方形を設定する
#define EM_SETSEL &HB1                      'エディットコントロール内部のテキストを選択する
#define EM_SETTABSTOPS &HCB                 'MLE内のタブストップを設定する
#define EM_SETWORDBREAKPROC &HD0            'エディットコントロール内で使うカスタムのワードブレーク文
                                            字を提供する
#define EM_UNDO &HC7                        'エディットコントロール内での直前の操作を取り消す

Var Shared Edit1 As Object
Var Shared Button1 As Object
Var Shared Text(10) As Object
Var Shared Radio(9) As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
For i = 0 To 10
    If i < 10 Then
        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 i

'=====================================
'=
'=====================================
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

    Text(0).SetWindowText ""

    Select Case Index
        Case 0                              'EM_CANUNDO
            Ret = Api_SendMessage(Edit1.GethWnd, EM_CANUNDO, 0, 0)
            If Ret = 0 Then
                Text(0).SetWindowText "UNDO不可"
            Else
                Text(0).SetWindowText "UNDO可"
            End If

        Case 1                              'EM_EMPTYUNDOBUFFER
            Ret = Api_SendMessage(Edit1.GethWnd, EM_EMPTYUNDOBUFFER, 0, 0)
```

```
        Case 2                                  'EM_GETFIRSTVISIBLELINE
            Ret = Api_SendMessage(Edit1.GethWnd, EM_GETFIRSTVISIBLELINE, 0, 0)
            Text(0).SetWindowText "表示最上行:" & Str$(Ret + 1) & "行目"

        Case 3                                  'EM_GETMODIFY
            Ret = Api_SendMessage(Edit1.GethWnd, EM_GETMODIFY, 0, 0)
            If Ret = 0 Then
                Text(0).SetWindowText "未修正"
            Else
                Text(0).SetWindowText "修正済"
            End If
        Case 4                                  'EM_SETMODIFY
            Ret = Api_SendMessage(Edit1.GethWnd, EM_SETMODIFY, 0, 0)

        Case 5                                  'EM_GETSEL
            Ret = Api_SendMessage(Edit1.GethWnd, EM_GETSEL, 0, 0)
            Text(0).SetWindowText "選択開始:" & Str$(Ret Mod (2 ^ 8)) & "～終了:" &
Str$(Int(Ret / (2 ^ 16))) & Chr$(13) & "レングス:" & Str$(Int(Ret / (2 ^ 16)) - Ret Mod (2 ^ 8))

        Case 6                                  'EM_LINELENGTH
            Ret = Api_SendMessage(Edit1.GethWnd, EM_LINELENGTH, -1, 0)
            Text(0).SetWindowText "行の未選択文字数:" & Str$(Ret)

        Case 7                                  'EM_SETSEL
            Ret = Api_SendMessage(Edit1.GethWnd, EM_SETSEL, 0, -1)

        Case 8                                  'EM_UNDO
            Ret = Api_SendMessage(Edit1.GethWnd, EM_UNDO, 0, 0)
            If Ret = 0 Then
                Text(0).SetWindowText "UNDO失敗"
            Else
                Text(0).SetWindowText "UNDO成功"
            End If

        Case 9                                  'EM_GETLINECOUNT
            Ret = Api_SendMessage(Edit1.GethWnd, EM_GETLINECOUNT, 0, 0)
            Text(0).SetWindowText Str$(Ret) & "行"
    End Select
End Sub

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

エディットボックスのスクロール情報

エディットボックスのスクロール情報を取得します。マルチライン(複数行入力あり)の場合のみ
GetScrollInfo スクロール情報取得

スクロール最大、最小および表示開始位置を表示させています。



```
'====================================
'= エディットボックスのスクロール情報
'=    (GetScrollInfo.bas)
'====================================
#include "Windows.bi"

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

' スクロール情報取得
Declare Function Api_GetScrollInfo& Lib "user32" Alias "GetScrollInfo" (ByVal hWnd&,
ByVal fBar&, lpScrollInfo As SCROLLINFO)

#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 SIF_ALL (&H1 Or &H2 Or &H4 Or &H10)  'SIF_RANGE Or SIF_PAGE Or SIF_POS Or
                                          SIF_TRACKPOS
#define SIF_DISABLENOSCROLL &H8           '無効なパラメータが指定されても消去せずに使用不能に
                                          する
#define SIF_PAGE &H2                      'nPageを設定することを明示
#define SIF_POS &H4                       'nPosを設定することを明示
#define SIF_RANGE &H1                     'nMinとnMaxを設定することを明示
#define SIF_TRACKPOS &H10                 'nTrackPosﾌｨｰﾙﾄﾞにつまみの位置をセットすることを明示
#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 text(5) As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
For i = 0 To 5
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1)))
    Text(i).SetFontSize 14
Next

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

    For i = 0 To 100
        txt = txt & Format$(i, "### ") & "http://tokovalue.jp" & vbCrLf
        Edit1.SetWindowText txt
    Next
End Sub

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

    si.cbSize = Len(si)
    si.fMask = SIF_ALL

    Ret = Api_GetScrollInfo(Edit1.GethWnd, SB_VERT, si)
    Text(3).SetWindowText Format$(si.nMax, "###")
    Text(4).SetWindowText Format$(si.nMin, "###")
```

```
    Text(5).SetWindowText Format$(si.nPos, "###")
End Sub

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

エディットボックスの読取専用設定・解除

SendMessage ウィンドウにメッセージを送信
EM_SETREADONLY(&HCF) エディットコントロールの読み取り専用スタイルを設定

起動時、通常入力(読取なし)に設定されています。
「読取専用」、「通常入力」をクリックし入力状態をチェックします。

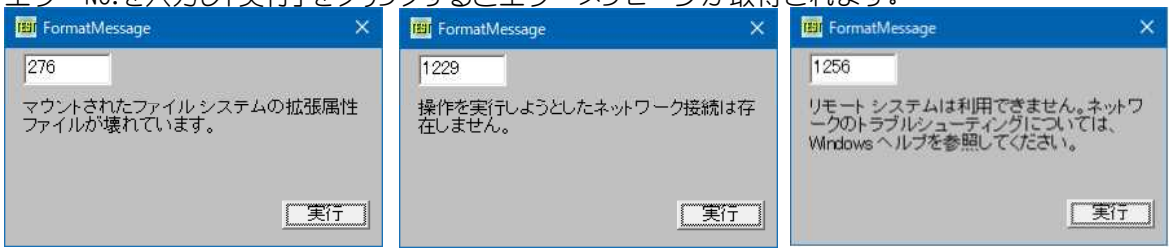
```
'=====================================
'= エディットボックスの読取専用設定・解除
'=    (EditBoxReadOnly.bas)
'=====================================
#include "Windows.bi"

' 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 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 EM_SETREADONLY &HCF                    'エディットコントロールの読み取り専用スタイルを設定する

Var Shared Edit1 As Object
Edit1.Attach GetDlgItem("Edit1")

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Edit1.SetFocus
End Sub

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

    Ret = Api_SendMessage(Edit1.GethWnd, EM_SETREADONLY, 1, 0)
    Edit1.SetSelText Len(Edit1.GetWIndowText), Len(Edit1.GetWIndowText)
    Edit1.SetFocus
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
```

```
Sub Button2_on()
    Var Ret As Long

    Ret = Api_SendMessage(Edit1.GethWnd, EM_SETREADONLY, 0, 0)
    Edit1.SetSelText Len(Edit1.GetWIndowText), Len(Edit1.GetWIndowText)
    Edit1.SetFocus
End Sub

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

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

エラーメッセージの取得

メッセージ文字列を書式化します。この関数は、入力としてメッセージ定義を受け取ります。
FormatMessage メッセージの文字列を指定の書式で取得

エラーNo.を入力し「実行」をクリックするとエラーメッセージが取得されます。



VBの場合、Err.LastDllError でエラー番号を取得できますが、F-Basic ではどうするのでしょう？？？
ちなみに、GetLastErrorは使えないようです。(全て「0」が返る)

```
'=====================================
'= エラーメッセージの取得
'=     (Formatmessage.bas)
'=====================================
#include "Windows.bi"

' 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 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
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 Erro As Long
    Var Buffer As String
    Var Ret As Long

    Erro = Val(Edit1.GetWindowText)

    Buffer = Space$(256)
    Ret = Api_FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM, 0, Erro, 0, Buffer, Len(Buffer),
ByVal 0)

    Text1.SetWindowText Left$(Buffer, Ret)
End Sub

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

参考(日本語エラーメッセージ一覧)

0:操作は正常に終了しました。
1:不正な関数です。
2:指定されたファイルが見つかりません。
3:指定されたパスが見つかりません。
4:システムはファイルを開けません。
5:アクセスは拒否されました。
6:ハンドルが無効です。
7:記憶領域制御ブロックが壊れていました。
8:このコマンドを実行するのに十分な記憶領域がありません。
9:記憶領域制御ブロックのアドレスが無効です。
10:環境が不正です。
11:不正な形式のプログラムを読み込もうとしました。
12:アクセスコードが無効です。
13:データが無効です。
14:この操作を完了するのに十分な記憶領域がありません。
15:システムは指定されたドライブを見つけられません。
16:ディレクトリを削除できません。
17:システムはファイルを別のディスクドライブに移動できません。
18:これ以上ファイルはありません。
19:このメディアは書き込み保護されています。
20:システムは指定されたデバイスを見つけられません。
21:デバイスの準備ができていません。
22:デバイスはそのコマンドを認識できません。
23:データエラー(CRCエラー)
24:プログラムがコマンドを発行しましたが、コマンドの長さが正しくありません。
25:指定された領域またはトラックがディスク上に見つかりません。
26:指定されたディスクまたはフロッピーディスクにアクセスできません。
27:ドライブに、要求されたセクタが見つかりませんでした。
28:プリンタが用紙切れです。
29:指定されたデバイスに書き込めません。
30:システムは、指定されたデバイスから読み取れません。
31:システムに装着されたデバイスは動作していません。
32:ファイルが別の処理で使われているため、アクセスできません。
33:別の処理がファイルの一部をロックしているため、ファイルにアクセスできません。
34:ドライブに入っているフロッピーディスクは正しくありません。%2(ボリュームシリアル番号:%3)をドライブ%1に入れてください。
36:開かれている共有ファイルが多すぎます。
38:ファイルの終わりに達しました。
39:ディスクがいっぱいです。
50:ネットワーク要求はサポートされていません。
51:リモートコンピュータは利用できません。

52:ネットワーク上に重複する名前があります。
53:ネットワークパスが見つかりませんでした。
54:ネットワークがビジー状態です。
55:指定されたネットワークリソースまたはデバイスは、もう使えません。
56:ネットワークBIOSコマンドの制限値に達しました。
57:ネットワークアダプタハードウェアエラーが発生しました。
58:指定されたサーバーは、要求された操作を実行できません。
59:予期しないネットワークエラーが発生しました。
60:リモートアダプタに互換性がありません。
61:プリンタキューがいっぱいです。
62:サーバー上に、印刷待ちファイルを格納する領域がありません。
63:印刷待ちだったファイルが削除されました。
64:指定されたネットワーク名はもう利用できません。
65:ネットワークアクセスが拒否されました。
66:ネットワークリソースの種類が正しくありません。
67:ネットワーク名が見つかりません。
68:ローカルコンピュータネットワークアダプタカードの名前の制限を超えました。
69:ネットワークBIOSセッションが制限を超えました。
70:リモートサーバーが一時停止されたか、または起動処理中です。
71:このリモートコンピュータには、もう接続できません。コンピュータが受け入れられる接続の限界に達しています。
72:指定されたプリンタまたはディスクは、一時停止されています。
80:ファイルは存在します。
82:ディレクトリまたはファイルは作成できません。
83:INT24で失敗
84:この要求を処理する記憶領域がありません。
85:ローカルデバイス名は既に使われています。
86:指定されたネットワークパスワードは正しくありません。
87:パラメータが正しくありません。
88:ネットワーク上で書き込みエラーが発生しました。
89:システムは別のプロセスを同時に起動できません。
100:別のシステムセマフォを作成できません。
101:排他セマフォは、別のプロセスが所有しています。
102:セマフォが設定されており、閉じられません。
103:セマフォはもう一度設定することはできません。
104:割り込み時に排他セマフォを要求できません。
105:セマフォの前回の所有権は終了しました。
106:ドライブ%1にフロッピーディスクを入れてください。
107:代わりのフロッピーディスクが挿入されなかったため、プログラムは停止しました。
108:ディスクは使用中か、または別の処理でロックされています。
109:パイプは終了しました。
110:システムは指定されたデバイスまたはファイルを開けません。
111:ファイル名が長すぎます。
112:ディスク上に十分な領域がありません。
113:利用できる内部ファイル識別子はこれ以上ありません。
114:対象の内部ファイル識別子が不正です。
117:アプリケーションによって出されたIOCTL呼び出しは正しくありません。
118:書き込み時検証スイッチのパラメータ値が正しくありません。
119:システムは要求されたコマンドをサポートしていません。
120:この関数はWin32モードでのみ有効です。
121:セマフォのタイムアウト期間に達しました。
122:システムコールに渡されたデータ領域が、小さすぎます。
123:ファイル名、ディレクトリ名、またはボリュームラベルの構文が正しくありません。
124:システムコールレベルが正しくありません。
125:ディスクにボリュームラベルがありません。
126:指定されたモジュールが見つかりませんでした。
127:指定されたプロシージャが見つかりませんでした。
128:待つべき子プロセスはありません。
129:%1アプリケーションはWin32モードでは実行できません。
130:rawディスクI/O以外の処理に、オープンディスクパーティションへのファイルハンドルを使おうとしました。
131:ファイルポインタをファイルの先頭よりも前に移動しようとしました。
132:ファイルポインタは指定されたデバイスまたはファイルに設定できません。
133:JOINまたはSUBSTコマンドは、以前に結合していたドライブを含むドライブには使えません。
134:既に結合させられているドライブに対して、JOINまたはSUBSTコマンドを実行しようとしました。
135:既に代替されているドライブに対して、JOINまたはSUBSTコマンドを実行しようとしました。
136:システムは結合していないドライブのJOINを削除しようとしました。
137:システムは代替されていないドライブの代替を削除しようとしました。
138:システムは、結合しているドライブ上のディレクトリに、ドライブを結合させようとしました。
139:システムは、代替されたドライブ上のディレクトリに、ドライブを代替しようとしました。
140:システムは、代替されたドライブ上のディレクトリに、ドライブを結合させようとしました。
141:システムは、結合しているドライブのディレクトリに、ドライブをSUBSTしようとしました。

142:システムは現在、JOINおよびSUBSTを実行できません。
143:システムは、同じドライブ上のディレクトリにドライブを結合または代替させることはできません。
144:ディレクトリはルートディレクトリのサブディレクトリではありません。
145:ディレクトリは空ではありません。
146:指定されたパスは代替で使われています。
147:リソース不足のため、このコマンドを実行できません。
148:指定されたパスは現在使えません。
149:ドライブ上のディレクトリが以前の代替の先であるドライブを結合または代替しようとしました。
150:システムトレース情報がCONFIG.SYSファイルで指定されていなかったか、またはトレースが許可されていません。
151:DosMuxSemWaitに指定されたセマフォイベントの数が、正しくありません。
152:DosMuxSemWaitは実行されませんでした。設定されているセマフォが多すぎます。
153:DosMuxSemWait一覧は正しくありません。
154:入力されたボリュームラベルの文字数は、対象側のファイルシステムでの文字数制限を超えています。
155:別のスレッドを作成できません。
156:受け側のプロセスがシグナルを拒否しました。
157:セグメントは既に破棄されており、ロックできません。
158:セグメントは既にロック解除されています。
159:スレッドIDのアドレスが正しくありません。
160:DosExecPgmに渡された引き数の文字列が、正しくありません。
161:指定されたパスは無効です。
162:既に保留にされているシグナルがあります。
164:システム内にこれ以上スレッドを作成できません。
167:ファイルの領域をロックできません。
170:要求したリソースは使用中です。
173:与えられた取り消し領域に対するロック要求は未解決ではありませんでした。
174:ファイルシステムは、ロックの種類に対するアトムレベルの変更をサポートしていません。
180:システムは不正なセグメント数を検出しました。
182:オペレーティングシステムは%1を実行できません。
183:既に存在するファイルを作成することはできません。
186:渡されたフラグは、正しくありません。
187:指定されたシステムセマフォ名が見つかりません。
188:オペレーティングシステムは%1を実行できません。
189:オペレーティングシステムは%1を実行できません。
190:オペレーティングシステムは%1を実行できません。
191:Win32モードでは%1を実行できません。
192:オペレーティングシステムは%1を実行できません。
193:%1は有効なWin32アプリケーションではありません。
194:オペレーティングシステムは%1を実行できません。
195:オペレーティングシステムは%1を実行できません。
196:オペレーティングシステムはこのアプリケーションを実行できません。
197:オペレーティングシステムは現在、このアプリケーションを実行するように設定されていません。
198:オペレーティングシステムは%1を実行できません。
199:オペレーティングシステムはこのアプリケーションを実行できません。
200:コードセグメントは64KB以下でなければいけません。
201:オペレーティングシステムは%1を実行できません。
202:オペレーティングシステムは%1を実行できません。
203:入力された環境オプションを見つけられませんでした。
205:コマンドサブツリーの中には、シグナルハンドラのあるプロセスはありません。
206:ファイル名または拡張子が長すぎます。
207:リング2スタックは使用中です。
208:ファイル名に文字*や?が不正に入力されたか、または、指定されたファイル名の文字が多すぎます。
209:ポストされたシグナルは正しくありません。
210:シグナルハンドラを設定できません。
212:セグメントはロックされており、再割り当てできません。
214:このプログラムまたはダイナミックリンクモジュールにアタッチされているダイナミックリンクモジュールが、多すぎます。
215:LoadModuleへの呼び出しを入れ子にすることはできません。
230:パイプの状態が無効です。
231:すべてのパイプインスタンスがビジー状態です。
232:パイプを閉じています。
233:パイプのもう一方の側には、プロセスはありません。
234:データがさらにあります。
240:セッションは取り消されました。
254:指定された拡張属性名は、無効です。
255:拡張属性が矛盾しています。
259:データはもうありません。
266:CopyAPIは使えません。
267:ディレクトリ名が無効です。
275:拡張属性がバッファにおさまりませんでした。

276:マウントされているファイルシステム上の拡張属性ファイルが、壊れています。
277:拡張属性テーブルファイルがいっぱいです。
278:指定された拡張属性ハンドルは無効です。
282:マウントされているファイルシステムは、拡張属性をサポートしていません。
288:呼び出し元が所有していないミューテックスを解放しようとしています。
298:1つのセマフォに対して作成したポストが多すぎます。
299:Read/WriteProcessMemory要求の一部のみが実行されました。
317:メッセージ番号0x%1のメッセージは、メッセージファイル%2にありません。
487:無効なアドレスにアクセスしようとしました。
534:演算結果が32ビットを超えました。
535:パイプのもう一方の端にプロセスがあります。
536:プロセスがパイプのもう一方の端を開くのを待っています。
994:拡張属性へのアクセスが拒否されました。
995:スレッドが終了したか、またはアプリケーション要求によって、I/O処理が中止されました。
996:重複するI/Oイベントは、シグナル状態ではありません。
997:重複するI/O処理が進行中です。
998:メモリの場所に無効なアクセスがありました。
999:インページ操作中にエラーが発生しました。
1001:再帰が深すぎて、スタックがあふれました。
1002:送信済みのメッセージに対する処理はできません。
1003:この関数を完了できません。
1004:無効なフラグです。
1005:ボリューム内に認識可能なファイルシステムがありません。必要なファイルシステムドライバがすべて読み込まれているか、またボリュームが壊れていないかを確認してください。
1006:ファイルのボリュームが大幅に変更されたため、開いているファイルはもう有効ではありません。
1007:要求された処理は全画面表示モードでは実行できません。
1008:存在しないトークンを参照しようとしました。
1009:設定レジストリデータベースが壊れています。
1010:設定レジストリキーが無効です。
1011:設定レジストリキーを開けませんでした。
1012:設定レジストリキーを読み込めませんでした。
1013:設定レジストリキーに書き込めませんでした。
1014:ファイルの1つをログまたは代替コピーで回復させる必要がありました。正常に回復しました。
1015:レジストリが壊れています。レジストリデータを含むファイルのうち、構造が壊れているものがあるか、または、メモリ中にあるファイルのシステムイメージが壊れているか、または、代替コピーおよびログが見つからないか壊れているため、ファイルを復元できませんでした。
1016:レジストリによるI/O処理が失敗し、復旧できない状態です。レジストリのシステムイメージを含むファイルを、読み取り、書き出し、またはフラッシュできませんでした。
1017:システムがレジストリにファイルを読み込みまたは復元しようとしましたが、指定されたファイルはレジストリファイル形式ではありません。
1018:削除の印が付けられたレジストリキーに対して、違法な操作を実行しようとしました。
1019:システムは、要求された領域をレジストリログ中に割り当てることができませんでした。
1020:既にサブキーや値を持っているレジストリキーには、シンボリックリンクは作成できません。
1021:一時的な親キーの下に、常設のサブキーは作成できません。
1022:変更通知要求は完了し、情報は呼び出し元のバッファに返されていません。呼び出し元で、ファイルを列挙し変更を見つける必要があります。
1051:実行中のほかのサービスが依存しているサービスに対して、停止コントロールが送信されました。
1052:要求された制御はこのサービスには無効です。
1053:開始または制御要求に対して、サービスは正しいタイミングで応答しませんでした。
1054:このサービスにスレッドを作成できませんでした。
1055:サービスデータベースはロックされています。
1056:サービスのインスタンスは既に実行されています。
1057:アカウント名が無効か、または存在しません。
1058:指定されたサービスは使用、または開始できません。
1059:循環するサービス依存関係が指定されました。
1060:指定されたサービスは、インストール済みのサービスとしては存在しません。
1061:現在、サービスは制御メッセージを受け入れられません。
1062:サービスは開始されていません。
1063:サービスプロセスはサービスコントローラに接続できませんでした。
1064:制御要求を処理中に、サービスで例外が発生しました。
1065:指定されたデータベースは存在しません。
1066:そのサービスはサービス固有のエラーコードを返しました。
1067:処理が予期せず終了しました。
1068:依存関係サービスまたはグループを開始できませんでした。
1069:ログオンエラーのため、サービスは開始しませんでした。
1070:サービス開始後、開始保留の状態で停止しました。
1071:指定されたサービスデータベースロックは無効です。
1072:指定されたサービスには削除の印が付いています。
1073:指定されたサービスは既に存在します。
1074:システムは現在、最後に認識された正常な設定で実行しています。

1075:依存サービスは存在しないか、または削除の印が付けられています。
1076:現在の起動は、既に最後に認識された正常な設定として受け入れられています。
1077:前回の起動以降、サービスは開始されていません。
1078:サービス名またはサービス表示名で、既に同じ名前が使われています。
1100:テープの物理的な終わりに達しました。
1101:テープアクセスがファイルマークに達しました。
1102:テープまたはパーティションの先頭に達しました。
1103:テープアクセスがファイルセットの終わりに達しました。
1104:テープ上にこれ以上データはありません。
1105:テープをパーティションに分割することはできませんでした。
1106:複数ボリュームパーティションの新しいテープにアクセスする場合、現在のブロックサイズは正しくありません。
1107:テープの読み込み時に、パーティション情報が見つかりませんでした。
1108:メディア取り出し機構をロックできません。
1109:メディアをアンロードできません。
1110:ドライブのメディアは変更されている可能性があります。
1111:I/Oバスがリセットされました。
1112:ドライブにメディアがありません。
1113:ターゲットのマルチバイトコードページに、Unicode文字へのマップが存在しません。
1114:ダイナミックリンクライブラリ(DLL)初期化ルーチンが失敗しました。
1115:システムのシャットダウンを行っています。
1116:シャットダウン中ではないので、シャットダウンは中止できません。
1117:I/Oデバイスエラーのため、要求を実行できませんでした。
1118:シリアルデバイスを初期化できませんでした。シリアルドライバはアンロードします。
1119:ほかのデバイスと割り込み要求(IRQ)を共有していたデバイスを開けません。そのIRQを使用しているデバイス少なくとも1つ既に開いています。
1120:シリアルポートへのほかの書き込みによって、シリアルI/O処理が完了しました。
(IOCTL_SERIAL_XOFF_COUNTERは0に達しました。)
1121:タイムアウトのため、シリアルI/O処理が完了しました。(IOCTL_SERIAL_XOFF_COUNTERは0に達しませんでした。)
1122:フロッピーディスク上に、IDアドレスマークが見つかりませんでした。
1123:フロッピーディスクセクタIDフィールドとフロッピーディスクコントローラトラックアドレスが一致しません。
1124:フロッピーディスクコントローラで、フロッピーディスクドライバの認識できないエラーが発生しました。
1125:フロッピーディスクコントローラは矛盾する結果をレジスタに返しました。
1126:ハードディスクへのアクセス時に補正操作に失敗し、再試行にも失敗しました。
1127:ハードディスクへのアクセス時にディスク操作に失敗し、再試行にも失敗しました。
1128:ハードディスクへのアクセス時にディスクコントローラのリセットが必要でしたが、失敗しました。
1129:テープの物理的な終わりに達しました。
1130:このコマンドを処理するのに十分なサーバーの記憶領域がありません。
1131:デッドロックが発生する可能性のある状態が検出されました。
1132:指定されているベースアドレスまたはオフセットは、適切に整列されていません。
1140:システムの電源状態の変更は、別のアプリケーションまたはドライバから拒否されました。
1141:システムBIOSは、システムの電源状態の変更に失敗しました。
1150:指定されたプログラムには、新しいバージョンのWindowsが必要です。
1151:指定されたプログラムは、WindowsプログラムまたはMS-DOSプログラムではありません。
1152:指定されたプログラムは、複数のインスタンスを起動できません。
1153:指定されたプログラムは、以前のバージョンのWindows用に書かれています。
1154:このアプリケーションの実行に必要なライブラリファイルのうちの1つが壊れています。
1155:この処理に指定されているファイルには、アプリケーションが関連付けられていません。
1156:アプリケーションへコマンドを送信中に、エラーが発生しました。
1157:このアプリケーションの実行に必要なライブラリファイルのうちの1つが見つかりません。
1200:指定されたデバイス名は無効です。
1201:デバイスは現在接続されていませんが、この接続は記録されています。
1202:以前に記録されていたデバイスを復元しようとしました。
1203:どのネットワークプロバイダも指定されたネットワークパスを受け付けませんでした。
1204:指定されたネットワークプロバイダ名は無効です。
1205:ネットワーク接続プロファイルを開けません。
1206:ネットワーク接続プロファイルが壊れています。
1207:非コンテナを列挙することはできません。
1208:拡張エラーが発生しました。
1209:指定されたグループ名の形式が無効です。
1210:指定されたコンピュータ名の形式が無効です。
1211:指定されたイベント名の形式が無効です。
1212:指定されたドメイン名の形式が無効です。
1213:指定されたサービス名の形式が無効です。
1214:指定されたネットワーク名の形式が無効です。
1215:指定された共有名の形式が無効です。
1216:指定されたパスワードの形式が無効です。
1217:指定されたメッセージ名の形式が無効です。
1218:指定されたメッセージ送信先の形式が無効です。
1219:入力された内容は、既存の資格証明のセットと一致しません。

1220:ネットワークサーバーにセッションを確立しようとしましたが、サーバー上に既に確立されているセッションが多すぎます。
1221:ワークグループ名またはドメイン名が、ネットワーク上のほかのコンピュータで既に使用されています。
1222:ネットワークが存在しないか、または起動されていません。
1223:操作はユーザーによって取り消されました。
1224:ユーザーマップセクションが開いているファイル上では、要求された操作は実行できません。
1225:リモートシステムによって、ネットワーク接続が拒否されました。
1226:ネットワーク接続は正常に終了しました。
1227:ネットワークトランスポートエンドポイントには、既に関連付けられたアドレスがあります。
1228:アドレスはまだネットワークエンドポイントと関連付けられていません。
1229:存在しないネットワーク接続に対して操作を実行しようとしました。
1230:アクティブなネットワーク接続に対して、無効な操作を実行しようとしました。
1231:このトランスポートではリモートネットワークに到達できません。
1232:このトランスポートではリモートシステムに到達できません。
1233:リモートシステムはトランスポートプロトコルをサポートしていません。
1234:リモートシステム上のネットワークエンドポイントでは、起動しているサービスはありません。
1235:要求は中断されました。
1236:ネットワーク接続は、ローカルシステムによって中断されました。
1237:その操作は完了できませんでした。再実行してください。
1238:アカウントの最大接続数に達したので、サーバーへ接続できません。
1239:このアカウントに許可されていない時間帯に、ログインしようとしています。
1240:このアカウントは、このステーションからログインする権限がありません。
1241:要求された操作には、このネットワークアドレスは使用できません。
1242:そのサービスは既に登録されています。
1243:指定されたサービスは存在しません。
1244:ユーザーが認証されなかったため、要求された操作は実行されませんでした。
1245:ユーザーがネットワークにログオンしていないため、要求された処理は実行されませんでした。指定されたサービスは存在しません。
1246:進行中の作業の続行を呼び出し側に求める値が返されました。
1247:初期化を行おうとしましたが、既に初期化は完了しています。
1248:ローカルデバイスはこれ以上ありません。
1300:参照されている特権のうちの一部は呼び出し側に割り当てられていません。
1301:アカウント名とセキュリティIDとのマッピングが、いくつか完了していません。
1302:このアカウントには、システム領域の制限値は特に設定されていません。
1303:利用できる暗号化キーがありません。既知の暗号化キーが返されました。
1304:WindowsNTパスワードが複雑すぎるため、LANManagerパスワードに変換できません。返されたLANManagerパスワードは空の文字列です。
1305:改訂レベルが不明です。
1306:2つの改訂レベルに互換性がないことを示します。
1307:このセキュリティIDはこのオブジェクトの所有者として割り当てられていない可能性があります。
1308:このセキュリティIDは、オブジェクトの主グループとして割り当てられていない可能性があります。
1309:偽装トークンを操作しようとしましたが、このスレッドは現在クライアントを偽装していません。
1310:グループはまだ使用できる可能性があります。
1311:現在、ログオン要求を処理するサーバーがありません。
1312:指定されたログオンセッションは存在しません。既に終了している可能性があります。
1313:指定された特権は存在しません。
1314:クライアントは必要な特権を保有していません。
1315:指定された名前は正しい形式のアカウント名ではありません。
1316:指定されたユーザーは既に存在します。
1317:指定されたユーザーは存在しません。
1318:指定されたグループは既に存在します。
1319:指定されたグループは存在しません。
1320:指定されたユーザーアカウントが既にグループのメンバーであるか、または指定されたグループにメンバーが含まれているためグループを削除できません。
1321:指定されたユーザーアカウントは指定されたグループアカウントのメンバーではありません。
1322:最後の管理アカウントは、使用不可にしたり削除したりできません。
1323:パスワードを更新できません。現在のパスワードが間違っています。
1324:パスワードを更新できません。新しいパスワードに、使用できない値が含まれています。
1325:パスワード更新ルールに違反したため、パスワードを更新できません。
1326:ログオン失敗:ユーザー名が不明、またはパスワードが誤っています。
1327:ログオン失敗:ユーザーアカウントの制限です。
1328:ログオン失敗:アカウントログオン時間の制限違反です。
1329:ログオン失敗:ユーザーはこのコンピュータへログオンを許可されていません。
1330:ログオン失敗:指定されたアカウントパスワードの有効期間が切れています。
1331:ログオン失敗:アカウントは現在、無効です。
1332:アカウント名とセキュリティIDとのマップがありません。
1333:一度に要求されたローカルユーザー識別子(LUID)が多すぎます。
1334:これ以上、ローカルユーザー識別子(LUID)は利用できません。
1335:セキュリティIDのサブオーソリティ部分は、この使用には無効です。
1336:アクセス制御リスト(ACL)の構造が無効です。

1337:セキュリティID構造体が無効です。
1338:セキュリティ記述子の構造体が無効です。
1340:継承されたアクセス制御リスト(ACL)またはアクセス制御エントリ(ACE)は、構築できませんでした。
1341:サーバーは現在、使用不可になっています。
1342:サーバーは現在、使用可能になっています。
1343:与えられた値は、識別子オーソリティには無効な値です。
1344:メモリ不足のため、セキュリティ情報の更新ができません。
1345:指定された属性が無効か、またはグループ全体の属性と矛盾します。
1346:必要な偽装レベルが与えられなかったか、または、与えられた偽装が無効です。
1347:匿名レベルのセキュリティトークンは開けません。
1348:要求された検証情報クラスが無効でした。
1349:この種類のトークンはこの使用に不適切です。
1350:関連付けられたセキュリテイがないオブジェクトでは、セキュリティ処理はできません。
1351:WindowsNTServerにアクセスできなかったか、または、ドメイン内のオブジェクトが保護されており、必要な情報を取得できませんでした。
1352:セキュリティアカウントマネージャ(SAM)またはローカルセキュリティオーソリティ(LSA)サーバーの状態が正しくないため、セキュリティ操作を行えませんでした。
1353:ドメインの状態が正しくないため、セキュリティ操作を行えませんでした。
1354:この操作はドメインのプライマリドメインコントローラにだけ許可されています。
1355:指定されたドメインは存在しません。
1356:指定されたドメインは既に存在します。
1357:サーバーあたりのドメイン数制限を超えようとしました。
1358:メディアのエラーか、またはディスク上のデータ構造体が壊れたために、要求した処理を完了できません。
1359:セキュリティアカウントデータベースに、内部的な矛盾があります。
1360:既に非ジェネリックタイプにマップされているはずのアクセスマスクに、ジェネリックアクセスタイプが含まれていました。
1361:セキュリティ記述子が正しい形式(完全または自己相関)ではありません。
1362:要求された処理は、ログオン処理のみに使用するものです。呼び出し処理はログオン処理として登録していません。
1363:既に使われているIDでは、新しいログオンセッションを開始できません。
1364:指定された認証パッケージは不明です。
1365:ログオンセッションは、要求した処理とは矛盾する状態にあります。
1366:ログオンセッションIDは既に使われています。
1367:ログオン要求には、無効なログオンタイプの値が含まれています。
1368:データがそのパイプから読み取られるまでは、その名前付きパイプ経由で偽装できません。
1369:レジストリサブツリーの処理状態は、要求した処理と矛盾します。
1370:内部セキュリティデータベースが壊れています。
1371:ビルトインアカウントには、この操作は実行できません。
1372:特殊なビルトイングループには、この操作は実行できません。
1373:特殊なビルトイングループには、この操作は実行できません。
1374:現在、このグループはユーザーの主グループなので、ユーザーは削除できません。
1375:トークンは主トークンとして既に使われています。
1376:指定されたローカルグループは存在しません。
1377:指定されたアカウント名は、ローカルグループのメンバーではありません。
1378:指定されたアカウント名は、既にローカルグループのメンバーです。
1379:指定されたローカルグループは既に存在します。
1380:ログオン失敗:ユーザーは要求したログオンの種類をこのコンピュータで得ていません。
1381:1つのシステムに保存できる機密の最大数を超えました。
1382:機密の長さが、最大許容値を超えました。
1383:ローカルセキュリティオーソリティデータベースに、内部矛盾があります。
1384:ログオンの間に、ユーザーのセキュリティコンテキストが蓄積したセキュリティIDが多すぎます。
1385:ログオン失敗:ユーザーは要求したログオンの種類をこのコンピュータで得ていません。
1386:ユーザーパスワードを変更するには、相互暗号化パスワードが必要です。
1387:メンバーが存在しないため、新しいメンバーをローカルグループに追加できませんでした。
1388:メンバーのアカウントタイプが違うため、新しいメンバーをローカルグループに追加できませんでした。
1389:指定されたセキュリティIDが多すぎます。
1390:このユーザーパスワードを変更するには、相互暗号化パスワードが必要です。
1391:ACLには継承コンポーネントは含まれていません。
1392:ファイルまたはディレクトリが壊れているので、読み取れません。
1393:ディスク構造体が壊れており、読み取ることができません。
1394:指定されたログオンセッションには、ユーザーセッションキーがありません。
1395:アクセス中のサービスは、特定の接続数に限定されています。現在、このサービスに接続できません。サービスが受け付けられる接続の最大数に達しています。
1400:無効なウィンドウハンドルです。
1401:無効なメニューハンドルです。
1402:無効なカーソルハンドルです。
1403:無効なアクセラレータテーブルハンドルです。
1404:無効なフックハンドルです。
1405:複数ウィンドウ位置構造体のハンドルが無効です。
1406:最上位の子ウィンドウを作成できません。

1407:ウィンドウクラスが見つかりません。
1408:無効なウィンドウです。ほかのスレッドに属しています。
1409:そのホットキーは既に登録されています。
1410:クラスは既に存在します。
1411:クラスが存在しません。
1412:クラスにはまだ開いているウィンドウがあります。
1413:無効なインデックスです。
1414:無効なアイコンハンドルです。
1415:プライベートDIALOGウィンドウワードを使っています。
1416:リストボックスの識別子が見つかりませんでした。
1417:ワイルドカードが見つかりませんでした。
1418:スレッドには、開いているクリップボードはありません。
1419:ホットキーは登録されていません。
1420:このウィンドウは、有効なダイアログウィンドウではありません。
1421:コントロールIDが見つかりません。
1422:エディットコントロールがないため、コンボボックスのメッセージが無効です。
1423:このウィンドウはコンボボックスではありません。
1424:高さは256未満でなければいけません。
1425:無効なデバイスコンテキスト(DC)ハンドルです。
1426:無効なフックプロシージャの種類です。
1427:無効なフックプロシージャです。
1428:モジュールハンドルなしでは、非ローカルフックを設定できません。
1429:このフックプロシージャは、グローバルにしか設定できません。
1430:ジャーナルフックプロシージャは既にインストールされています。
1431:フックプロシージャはインストールされていません。
1432:単一選択のリストボックスには、無効なメッセージです。
1433:LB_SETCOUNTがnon-lazyリストボックスに送られました。
1434:このリストボックスは、タブ停止をサポートしません。
1435:ほかのスレッドで作成されたオブジェクトは壊せません。
1436:子ウィンドウには、メニューを添付できません。
1437:ウィンドウにシステムメニューがありません。
1438:メッセージボックスのスタイルが無効です。
1439:無効なシステム全体に渡る(SPI_*)パラメータです。
1440:画面は既にロックされています。
1441:複数ウィンドウ位置構造体のウィンドウへのハンドルは、すべて親が同じでなければいけません。
1442:このウィンドウは子ウィンドウではありません。
1443:無効なGW_*コマンドです。
1444:無効なスレッド識別子です。
1445:マルチドキュメントインターフェイス(MDI)ウィンドウでないウィンドウからは、メッセージを処理できません。
1446:ポップアップメニューは既にアクティブになっています。
1447:このウィンドウにスクロールバーはありません。
1448:スクロールバーの範囲は0x7FFF以下でなければなりません。
1449:指定された方法でウィンドウを表示または削除できません。
1450:システムリソースが足りないため、要求されたサービスを完了できません。
1451:システムリソースが足りないため、要求されたサービスを完了できません。
1452:システムリソースが足りないため、要求されたサービスを完了できません。
1453:割り当て領域が不足しているため、要求されたサービスを完了できません。
1454:割り当て領域が不足しているため、要求されたサービスを完了できません。
1455:ページングファイルが小さすぎるため、この操作を完了できません。
1456:メニュー項目が見つかりませんでした。
1500:イベントログファイルが壊れています。
1501:イベントログファイルが開けなかったので、イベントログサービスは開始しませんでした。
1502:イベントログファイルがいっぱいです。
1503:読み取りと読み取りの間にイベントログファイルが変更されました。
1700:文字列のバインドが無効です。
1701:バインドハンドルの種類が正しくありません。
1702:バインドハンドルが無効です。
1703:RPCプロトコルシーケンスはサポートされていません。
1704:RPCプロトコルシーケンスが無効です。
1705:文字列のユニバーサル一意識別子(UUID)が無効です。
1706:エンドポイントの形式が無効です。
1707:ネットワークアドレスが無効です。
1708:エンドポイントが見つかりません。
1709:タイムアウトの値が無効です。
1710:オブジェクトのユニバーサル一意識別子(UUID)を見つけることができませんでした。
1711:オブジェクトのユニバーサル一意識別子(UUID)は既に登録されています。
1712:タイプのユニバーサル一意識別子(UUID)は既に登録されています。
1713:RPCサーバーは既に待機中です。
1714:プロトコルシーケンスが登録されていません。
1715:RPCサーバーは応答しません。

1716:マネージャの種類が不明です。
1717:インターフェイスが不明です。
1718:バインドはありません。
1719:プロトコルシーケンスはありません。
1720:エンドポイントを作成できません。
1721:リソースが足りないため、この操作を完了できません。
1722:RPCサーバーは使えません。
1723:RPCサーバーがビジー状態のため、この操作を完了できません。
1724:ネットワークオプションが無効です。
1725:スレッド中に、アクティブなリモートプロシージャコールはありません。
1726:リモートプロシージャコールに失敗しました。
1727:リモートプロシージャコールに失敗し、実行されませんでした。
1728:リモートプロシージャコール(RPC)プロトコルエラーが発生しました。
1730:その転送の構文はRPCサーバーでサポートされていません。
1732:このユニバーサル一意識別子(UUID)の種類はサポートされていません。
1733:タグが無効です。
1734:配列の範囲が無効です。
1735:バインドには、エントリ名は含まれていません。
1736:名前の構文が無効です。
1737:その名前の構文はサポートされていません。
1739:使用できるネットワークアドレスがないため、ユニバーサル一意識別子(UUID)を作成できません。
1740:エンドポイントが重複しています。
1741:認証の種類が不明です。
1742:呼び出しの最大数が小さすぎます。
1743:文字列が長すぎます。
1744:RPCプロトコルシーケンスが見つかりませんでした。
1745:プロシージャ番号が範囲外です。
1746:バインドには、認証情報は含まれていません。
1747:認証サービスが不明です。
1748:認証のレベルが不明です。
1749:セキュリティコンテキストが無効です。
1750:認可サービスが不明です。
1751:エントリが無効です。
1752:サーバーエンドポイントは、操作を実行できません。
1753:エンドポイントマッパーには、これ以上エンドポイントはありません。
1754:インターフェイスがエクスポートされていません。
1755:エントリ名が不完全です。
1756:バージョンオプションが無効です。
1757:メンバーはこれ以上ありません。
1758:アンエクスポートするものはありません。
1759:インターフェイスが見つかりませんでした。
1760:このエントリは既に存在します。
1761:エントリが見つかりません。
1762:名前のサービスが利用できません。
1763:ネットワークアドレスファミリが無効です。
1764:要求された操作はサポートされません。
1765:偽装を許可するセキュリティコンテキストはありません。
1766:リモートプロシージャコール(RPC)で、内部エラーが発生しました。
1767:RPCサーバーで0による整数除算を実行しようとしました。
1768:RPCサーバーでアドレス指定エラーが発生しました。
1769:RPCサーバーの浮動小数点演算で0による除算が実行されました。
1770:RPCサーバーで、浮動小数点の下位桁あふれが起きました。
1771:RPCサーバーで、浮動小数点の桁あふれが起きました。
1772:自動ハンドルのバインド用のRPCサーバーの一覧はもうありません。
1773:文字翻訳テーブルファイルを開けません。
1774:文字翻訳テーブルを含むファイルが、512バイト未満です。
1775:リモートプロシージャコールの間に、NULLコンテキストハンドルがクライアントからホストへ渡されました。
1777:リモートプロシージャコールの間に、コンテキストハンドルが変更されました。
1778:リモートプロシージャコールに渡されたバイントハンドルは、一致しません。
1779:スタブが、リモートプロシージャコールハンドルを取得できません。
1780:NULL参照ポインタがスタブに渡されました。
1781:数値が範囲外です。
1782:バイト数が小さすぎます。
1783:スタブが不良データを受け取りました。
1784:与えられたユーザーバッファは、要求した処理には無効です。
1785:ディスクが認識されません。フォーマットされていない可能性があります。
1786:このコンピュータにはセキュリティ情報はありません。
1787:WindowsNTサーバー上のSAMデータベースには、このセキュリティ設定のコンピュータアカウントはありません。
1788:主ドメインと被トラストドメインとのトラスト関係が、壊れました。

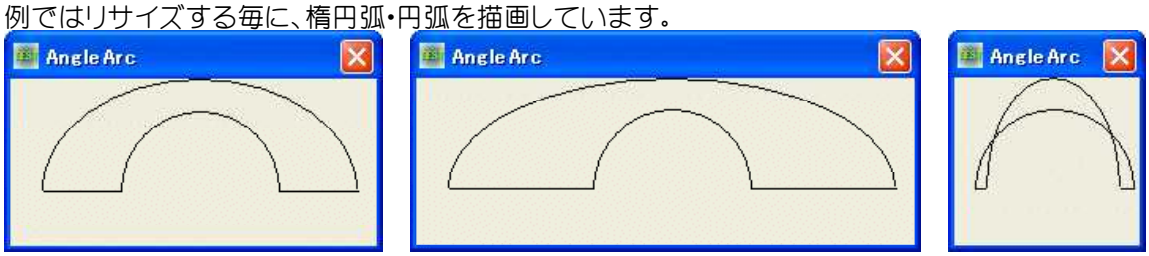
1789:このワークステーションと主ドメインとのトラスト関係が、壊れました。
1790:ネットワークログオンエラー
1791:このスレッドのリモートプロシージャコールは既に実行中です。
1792:ログオンしようとしましたが、ネットワークログオンサービスが開始されませんでした。
1793:ユーザーのアカウントの有効期限が切れました。
1794:リダイレクタは使用中なので、アンロードできません。
1795:指定されたプリンタドライバは既にインストールされています。
1796:指定されたポートは不明です。
1797:プリンタドライバが不明です。
1798:印刷プロセッサが不明です。
1799:指定された区切りファイルは無効です。
1800:指定された優先順位は無効です。
1801:プリンタ名が無効です。
1802:プリンタは既に存在します。
1803:プリンタコマンドが無効です。
1804:指定されたデータ型が無効です。
1805:指定された環境が無効です。
1806:バインドはこれ以上ありません。
1807:使用されているアカウントは、ドメイン内トラストアカウントです。このサーバーへのアクセスには、グローバルユーザーアカウントか、ローカルユーザーアカウントを使ってください。
1808:使用されているアカウントは、コンピュータアカウントです。このサーバーへのアクセスには、グローバルユーザーアカウントか、ローカルユーザーアカウントを使ってください。
1809:使用されているアカウントは、サーバートラストアカウントです。このサーバーへのアクセスには、グローバルユーザーアカウントか、ローカルユーザーアカウントを使ってください。
1810:指定されたドメインの名前、またはセキュリティID(SID)が、ドメインのトラスト情報と一致しません。
1811:サーバーは使用中のため、アンロードできません。
1812:指定されたイメージファイルには、リソースセクションがありませんでした。
1813:指定されたリソースの種類は、イメージファイルに見つかりません。
1814:指定されたリソース名は、イメージファイルに見つかりません。
1815:指定されたリソース言語IDは、イメージファイルに見つかりません。
1816:割り当て領域が不足のため、このコマンドを実行できません。
1817:登録されているインターフェイスはありません。
1818:この呼び出しの処理中に、サーバーが変更されました。
1819:バインドハンドルには、必要な情報が足りません。
1820:通信に失敗しました。
1821:要求された認証レベルは、サポートされていません。
1822:主要名が登録されていません。
1823:指定されたエラーは、有効なWindowsRPCエラーコードではありません。
1824:このコンピュータ上でのみ有効なUUIDが割り当てられました。
1825:セキュリティパッケージ固有のエラーが発生しました。
1826:スレッドは取り消されていません。
1827:ハンドルのエンコード/デコードで無効な処理が行われました。
1828:連続パッケージの互換性のないバージョンです。
1829:互換性のないバージョンのRPCスタブです。
1898:グループメンバーが見つかりませんでした。
1899:エンドポイントマッパーデータベースは作成できませんでした。
1900:オブジェクトのユニバーサル一意識別子(UUID)は存在しないUUIDです。
1901:指定された時刻は無効です。
1902:指定された用紙名は無効です。
1903:指定された用紙サイズは無効です。
1904:指定されたプリンタハンドルは、既に待ち状態です。
1905:指定されたプリンタは削除されました。
1906:プリンタの状態が無効です。
1907:ユーザーは、最初にログオンする前にパスワードを変更しなければなりません。
1908:このドメインのドメインコントローラが見つかりませんでした。
1909:参照されているアカウントは現在ロックされており、ログオンできない可能性があります。
2000:ピクセル形式が無効です。
2001:指定されたドライバは無効です。
2002:この操作に対してウィンドウスタイルまたはクラス属性が無効です。
2003:要求されたメタファイル処理は、サポートされていません。
2004:要求された変換操作は、サポートされていません。
2005:要求されたクリップ処理は、サポートされていません。
2202:指定されたユーザー名は無効です。
2250:このネットワーク接続は存在しません。
2401:このネットワーク接続には、開いているファイルまたは保留中の要求があります。
2402:アクティブな接続がまだあります。
2404:デバイスはアクティブな処理で使われており、切断できません。
3000:指定された印刷モニタは不明です。
3001:指定されたプリンタドライバは現在使用されています。
3002:スプールファイルが見つかりませんでした。

3003:StartDocPrinterコールが発行されませんでした。
3004:AddJobコールが発行されませんでした。
3005:指定された印刷プロセッサは既にインストールされています。
3006:指定された印刷モニタは既にインストールされています。
4000:コマンドの実行中に、WINSでエラーが発生しました。
4001:ローカルWINSを削除できません。
4002:ファイルからの取り込みに失敗しました。
4003:バックアップに失敗しました。以前に一括バックアップをしたことがありますか?
4004:バックアップに失敗しました。データベースをバックアップしているディレクトリを調べてください。
4005:WINSデータベースには名前がありません。
4006:設定されていないパートナーとの応答は、許可されていません。
6118:このワークグループのサーバー一覧は、現在使えません。
197120:基礎ファイルが複合ファイル形式に変換されました。
262144:要求した情報を得るには、レジストリデータベースを使ってください。
262145:成功しましたが、静的です。
262146:Macintoshクリップボード形式
262400:ドロップ゜は正しく実行されました。
262401:ドラッグアンドドロップ゜操作は取り消されました。
262402:既定のカーソルを使ってください。
262448:データには同じFORMATETCがあります。
262464:表示は既に固定されています。
262512:FORMATETCはサポートされていません。
262513:同一キャッシュ
262514:一部のキャッシュが更新されていません。
262528:OLEオブジェクトに対して無効な動詞です。
262529:動詞番号は有効ですが、ここで実行できません。
262530:無効なウィンドウハンドルが渡されました。
262560:メッセージが長すぎます。表示するために、何文字かが切り捨てられました。
262592:OLESTREAMをIStorageに変換できません。
262626:モニカはそれ自身で縮小しました。
262628:共通のプレフィックスはこのモニカです。
262629:共通のプレフィックスは入力モニカです。
262630:共通のプレフィックスは両方のモニカです。
262631:モニカは実行中のオブジェクトテーブルに既に登録されています。

円弧を描画

円弧を描画します。
GetSysColorシステムの背景色を取得
AngleArc 中心点と半径を指定して円・円弧を描画
GetArcDirection 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
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

例ではリサイズする毎に、楕円弧・円弧を描画しています。




```
'=====================================
'= 円弧を描画
'=====================================
#include "Windows.bi"

Type POINTAPI
    x   As Long
    y   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&)
```

```
' 中心点と半径を指定して円・円弧を描画
Declare Function Api_AngleArc& Lib "gdi32" Alias "AngleArc" (ByVal hDC&, ByVal x&, ByVal
y&, ByVal dwRadius&, ByVal eStartAngle!, ByVal eSweepAngle!)

' 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&)

' 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&, lpPoint As POINTAPI)

' 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                    'コマンドボタンの表面色
#define AD_CLOCKWISE 2                      '時計回り
#define AD_COUNTERCLOCKWISE 1               '反時計回り

'=====================================
'=
'=====================================
Declare Sub MainForm_Resize edecl ()
Sub MainForm_Resize()
    Var rgbColor As Long
    Var PT As POINTAPI
    Var hDC As Long
    Var Ret As Long

    rgbColor = Api_GetSysColor(COLOR_BTNFACE)

    SetBackColor rgbColor
    Cls
    ShowWindow -1

    hDC = Api_GetDC(GethWnd)

    If APi_GetArcDirection(hDC) = AD_CLOCKWISE Then
        Ret = Api_SetArcDirection(hDC, AD_COUNTERCLOCKWISE)
    End If

    Ret = Api_MoveToEx(hDC, GetWidth - 20, GetHeight / 2, PT)

    Ret = Api_AngleArc(hDC, GetWidth / 2, GetHeight / 2, GetHeight / 2 - 20, 0, 180)

    Ret = Api_SetArcDirection(hDC, AD_CLOCKWISE)

    Ret = Api_ArcTo(hDC, 20, 0, GetWidth - 20, GetHeight, 0, GetHeight / 2, GetWidth,
GetHeight / 2)

    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

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

円周のみのフォーム

円周のみのフォームを作成します。
ReleaseCapture マウスキャプチャの解放
Sendmessage ウィンドウにメッセージを送信
CreateEllipticRgn 楕円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
CombineRgn 既存の二つの領域を結合して新しい領域を作成
SetWindowPos ウィンドウのサイズ、位置およびZオーダーを設定

例では、3ドットの楕円を描画し、円周のみをフォームとしています。
円周上をドラッグするとフォームを移動させることが出来ますが、それ以外は受けつけません。
同様に、円周上をダブルクリックで終了させています。

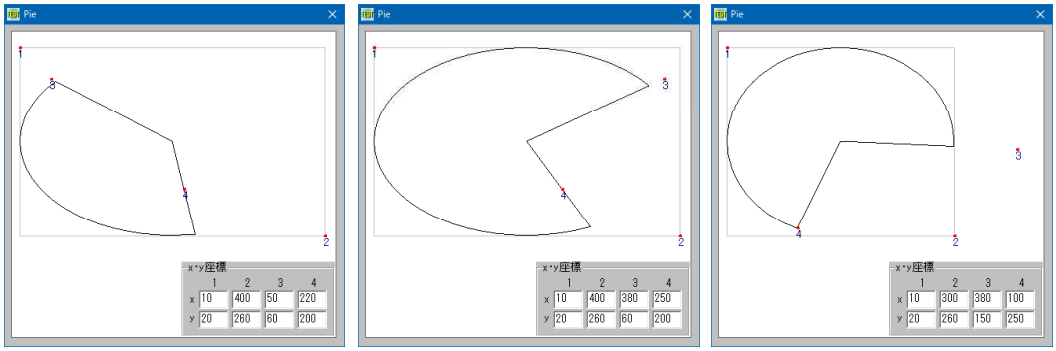

```
'=======================================
'= 円周のみのフォーム
'=    (CreateEllipticRgn2.bas)
'=======================================
#include "Windows.bi"

' 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。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)

' 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&)

' 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&)

' 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&)

' 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 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
#define RGN_DIFF 4                          'HRGNSRC1からHRGNSRC2を除いた領域
#define HWND_TOPMOST (-1)                   'ウィンドウを常に最前面に配置
#define SWP_SHOWWINDOW &H40                 'ウィンドウを表示する
#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

'=======================================
'=
'=======================================
Declare Sub ShapeForm()
Sub ShapeForm()
    Var iRgn As Long
    Var oRgn As Long
```

```
    Var Ret As Long

    SetBackColor RGB(255, 0, 0)
    Cls

    '楕円描画(外側)
    oRgn = Api_CreateEllipticRgn(0, 0, GetWidth, GetHeight)

    '楕円描画(内側)
    iRgn = Api_CreateEllipticRgn(3, 3, GetWidth - 3, GetHeight - 3)

    '新しい領域を作成
    Ret = Api_CombineRgn(oRgn, oRgn, iRgn, RGN_DIFF)

    '指定の領域をウィンドウ領域として設定
    Ret = Api_SetWindowRgn(GethWnd, oRgn, True)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_MouseDown edecl (Button As Integer, Shift As Integer, x As Single, y
As Single)
Sub MainForm_MouseDown(Button As Integer, Shift As Integer, x As Single, y As Single)
    Var Ret As Long

    Ret = Api_ReleaseCapture()
    Ret = Api_SendMessage(GethWnd, WM_NCLBUTTONDOWN, HTCAPTION, 0)
End Sub

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

    Ret = Api_SetWindowPos(GethWnd, HWND_TOPMOST, 0, 0, 0, 0, SWP_SHOWWINDOW Or SWP_NOMOVE
Or SWP_NOSIZE)
    ShapeForm
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_DblClick edecl ()
Sub MainForm_DblClick()
    End
End Sub

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

扇形の描画

Pie 扇形を描画する

例では、座標を変て描画される図形の確認をしています。扇を形成する円(楕円)は、•1と•2で表される矩形領域に内接します。扇の開始位置は、円(楕円)の中心と•3を結ぶ延長線上の交点、扇の終了位置は、中心と•4を結ぶ延長線上の交点です。



```
'=======================================
'= 扇形の描画
'=    (Pie.bas)
'=======================================
#include "Windows.bi"

' 扇形を描画する
Declare Function Api_Pie& Lib "gdi32" Alias "Pie" (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 Text(6) As Object
Var Shared Edit(7) As Object
Var Shared Picture1 As Object
Var Shared Group1 As Object
Var Shared Timer1 As Object

For i = 0 To 7
    If i < 6 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
Picture1.Attach GetDlgItem("Picture1")
Group1.Attach GetDlgItem("Group1") : Group1.SetFontSize 14
Timer1.Attach GetDlgItem("Timer1")

Var Shared hDC As Long

'=======================================
'=
'=======================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    ShowWindow -1
    Cls

    hDC = Api_GetDC(Picture1.GethWnd)

    Timer1.SetInterval 100
    Timer1.Enable -1
End Sub

'=======================================
'=
'=======================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var x(3) As Long
    Var y(3) As Long
    Var Ret As Long

    Picture1.Cls
```

```
    For i = 0 To 3
        x(i) = Val(GetDlgItemText("Edit" & Trim$(Str$(i + 1))))
        y(i) = Val(GetDlgItemText("Edit" & Trim$(Str$(i + 5))))
    Next

    Picture1.Line(x(0), y(0)) - (x(1), y(1)), , 14, b

    Ret = Api_Pie(hDC, x(0), y(0), x(1), y(1), x(2), y(2), x(3), y(3))

    For i = 0 To 3
        Picture1.SetDrawWidth 4
        Picture1.Pset(x(i), y(i)), 5
        Picture1.SetDrawWidth 0
        Picture1.Symbol(x(i) - 1, y(i) + 1), Trim$(Str$(i + 1)), 1, 1, 2
    Next
End Sub

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

    Ret = Api_ReleaseDC(Picture1.GethWnd, hDC)
    End
End Sub

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

オートコンプリート

SHAutoComplete オートコンプリートをエディットボックスに使うように指定
DllGetVersion Shell32.dllのバージョン取得


```
'=======================================
'= オートコンプリート
'=    (SHAutoComplete.bas)
'=======================================
#include "Windows.bi"

#define SHACF_DEFAULT &H0                   '(SHACF_FILESYSTEM Or SHACF_URLALL)
#define SHACF_FILESYSTEM &H1                '仮想フォルダをなどファイル一覧を含む
#define SHACF_URLHISTORY &H2                'ユーザの履歴内のURLを含む
#define SHACF_URLMRU &H4                    '最近使ったURLリスト内のURLを含む
#define SHACF_USETAB &H8                    'Tabキーを押すことにより自動補完リストから選択できる
#define SHACF_FILESYS_ONLY &H10
#define SHACF_URLALL (SHACF_URLHISTORY Or SHACF_URLMRU)
```

```
#define SHACF_AUTOSUGGEST_FORCE_ON  &H10000000
                                                'レジストリ値を無視し、自動補助機能をオンにする
#define SHACF_AUTOSUGGEST_FORCE_OFF &H20000000
                                                'レジストリ標準の値を無視し、自動補助機能をオフにする
#define SHACF_AUTOAPPEND_FORCE_ON &H40000000
                                                'レジストリ値を無視し、自動補完機能を強制的にオンにする
#define SHACF_AUTOAPPEND_FORCE_OFF -2147483648
                                                '標準の設定を無視し、自動補完機能を強制的にオフにする
#define S_OK &H0                                        '

#define DLLVER_PLATFORM_WINDOWS &H1         'Windows 95
#define DLLVER_PLATFORM_NT &H2              'Windows NT

Type DllVersionInfo
    cbSize              As Long
    dwMajorVersion      As Long             'メジャーバージョン
    dwMinorVersion      As Long             'マイナーバージョン
    dwBuildNumber       As Long             'ビルド番号
    dwPlatformID        As Long
End Type

' オートコンプリートをエディットボックスに使うように指定
Declare Function Api_SHAutoComplete& Lib "shlwapi" Alias "SHAutoComplete" (ByVal
hwndEdit&, ByVal dwFlags&)

' Shell32.dllのバージョン取得
Declare Function Api_DllGetVersion& Lib "shlwapi" Alias "DllGetVersion" (dwVersion As
DLLVERSIONINFO)
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 dvi As DLLVERSIONINFO

'=====================================
'=
'=====================================
Declare Function GetIEVersion(dvi As DLLVERSIONINFO) As Long
Function GetIEVersion(dvi As DLLVERSIONINFO) As Long
    Var Ret As Long

    dvi.cbSize = Len(dvi)
    Ret = Api_DllGetVersion(dvi)

    GetIEVersion = dvi.dwMajorVersion
End Function

'=====================================
'=
'=====================================
Declare Function GetIEVersionString() As String
Function GetIEVersionString() As String
    Var Ret As Long

    dvi.cbSize = Len(dvi)
    Ret = Api_DllGetVersion(dvi)

    GetIEVersionString = "IE Version:" & Trim$(Str$(dvi.dwMajorVersion)) & "." &
Trim$(Str$(dvi.dwMinorVersion)) & "." & Trim$(Str$(dvi.dwBuildNumber))
End Function

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

```
    Text1.SetWindowText GetIEVersionString

    Button1.SetWindowText "オートコンプリート Off"
    Edit1.SetFocus
End Sub

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

    If GetIEVersion(dvi) >= 5 Then
        Ret = Api_SHAutoComplete(Edit1.GethWnd, SHACF_URLHISTORY Or SHACF_URLMRU)
        Button1.SetWindowText "オートコンプリート On"
        Button1.EnableWindow 0
        Edit1.SetFocus

        Edit1.SetSelText 0, -1
    Else
        A% = MessageBox("", "IE5以上が必要です", 0, 2)
    End If
End Sub

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

大文字のパスを小文字に変換

PathMakePretty 大文字のパスを小文字に変換


```
'========================================
'= 大文字のパスを小文字に変換
'=    (PathMakePretty.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$)

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 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

    Ret = Api_PathMakePretty(sSave)

    Text1.SetWindowText sSave
End Sub

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

お気に入りに追加と整理

お気に入りに追加およびお気に入りの整理ダイアログを呼び出します。
DoAddToFavDlg お気に入りに追加
DoOrganizeFavDlg お気に入りの整理
SHGetFolderPath CSIDLからパスを取得
SHGetSpecialFolderLocation 特殊フォルダのパスを取得
WritePrivateProfileString iniファイルに文字列を書き込む
CoTaskMemFree アイテムIDリストの解放

お気に入りに追加をクリック

お気に入りの整理をクリック


```
'=====================================
'= お気に入りに追加と整理
'=    (DoAddToFavDlg.bas)
'=====================================
#include "Windows.bi"
```

```
#define MAX_PATH 260
#define ERROR_Success 0                         '正常終了の戻り値を示す
#define S_OK 0
#define S_FALSE 1
#define SHGFP_TYPE_CURRENT 0                    'フォルダの現在のパス名を返す
#define SHGFP_TYPE_DEFAULT 1                    'フォルダのデフォルトのパス名を返す
#define CSIDL_FAVORITES &H6                     'お気に入り(ファイルシステムディレクトリ)

' お気に入りに追加
Declare Function Api_DoAddToFavDlg& Lib "shdocvw" Alias "DoAddToFavDlg" (ByVal hWnd&,
ByVal Path$, ByVal PathSize&, ByVal Title$, ByVal TitleSize&, ByVal pidl&)

' お気に入りの整理
Declare Function Api_DoOrganizeFavDlg& Lib "shdocvw" Alias "DoOrganizeFavDlg" (ByVal
hWnd&, ByVal RootFolder$)

' CSIDLからパスを取得する関数
Declare Function Api_SHGetFolderPath& Lib "ShFolder" Alias "SHGetFolderPathA" (ByVal
hwndOwner&, ByVal nFolder&, ByVal hToken&, ByVal dwFlags&, ByVal pPath$)

' 特殊フォルダのパスを取得。「スタートメニュー」「送る」「最近使ったファイル」
Declare Function Api_SHGetSpecialFolderLocation& Lib "shell32" Alias
"SHGetSpecialFolderLocation" (ByVal hwndOwner&, ByVal nFolder&, pidl&)

' iniファイルに文字列を書き込む
Declare Function Api_WritePrivateProfileString& Lib "Kernel32" Alias
"WritePrivateProfileStringA" (ByVal lpApplicationName$, ByVal KeyName As Any, ByVal
lpString As Any, ByVal lpFileName$)

' アイテムIDリストの解放
Declare Sub Api_CoTaskMemFree Lib "ole32" Alias "CoTaskMemFree" (ByVal hMem&)

Var Shared Edit(2) As Object
Var Shared Text(2) As Object
Var Shared Button(1) As Object

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

'=====================================
'= お気に入りに追加(処理)
'=====================================
Declare Function AddFavourite(Title As String, Url As String) As String
Function AddFavourite(Title As String, Url As String) As String
    Var Success As Long
    Var ePos As Long
    Var PathSize As Long
    Var TitleSize As Long
    Var pidl As Long
    Var Path As String
    Var Ret As Long

    Title = Title & Chr$(0)
    TitleSize = Len(Title)

    Path = Space$(MAX_PATH) & Chr$(0)
    PathSize = Len(Path)

    If Api_SHGetSpecialFolderLocation(GethWnd, CSIDL_FAVORITES, pidl) = ERROR_Success
Then
        Success = Api_DoAddToFavDlg(GethWnd, Path, PathSize, Title, TitleSize, pidl)

        If Success = 1 Then
            ePos = InStr(Path, Chr$(0))
```

```
            Path = Left$(Path, ePos - 1)

            ePos = InStr(Title, Chr$(0))
            Title = Left$(Title, ePos - 1)

            Edit(0).SetWindowText Path
            Edit(1).SetWindowText Title

            AddFavourite = Path
        End If

        Api_CoTaskMemFree pidl
    End If
End Function

'=====================================
'=
'=====================================
Declare Sub SaveItem(SectionName As String, KeyName As String, Value As String, iniFile As
String)
Sub SaveItem(SectionName As String, KeyName As String, Value As String, iniFile As String)
    Var Ret As Long

    Ret = Api_WritePrivateProfileString(SectionName, KeyName, Value, iniFile)
End Sub

'=====================================
'=
'=====================================
Declare Function GetFolderPath(CSIDL As Long) As String
Function GetFolderPath(CSIDL As Long) As String
    Var Path As String
    Var Tmp As String

    Path = Space$(MAX_PATH)

    If Api_SHGetFolderPath(GethWnd, CSIDL, 0, SHGFP_TYPE_CURRENT, Path) = S_OK Then
        GetFolderPath = Left$(Path, InStr(Path, Chr$(0)) - 1)
    End If
End Function

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Edit(0).SetWindowText "F-Basic Programming Tips"
    Edit(1).SetWindowText "http://tokovalue.hp.infoseek.co.jp"
    Edit(2).SetWindowText ""
    ShowWindow -1
End Sub

'=====================================
'= お気に入りに追加
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Title As String
    Var Url As String
    Var Ret As String

    Title = Edit(0).GetWindowText

    Url = Edit(1).GetWindowText
    Ret = AddFavourite(Title, Url)

    Edit(0).SetWindowText Title
    Edit(1).SetWindowText Url
    Edit(2).SetWindowText Ret
End Sub
```

```
'=====================================
'= お気に入りの整理
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var RootFolder As String
    Var Success As Long

   RootFolder = GetFolderPath(CSIDL_FAVORITES)
   Success = Api_DoOrganizeFavDlg(GethWnd, RootFolder)
End Sub

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

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

親ウィンドウを変更する

SetParent 指定された子ウィンドウの親ウィンドウを変更

エディットボックスの親を、自フォーム<->デスクトップと変更しています。

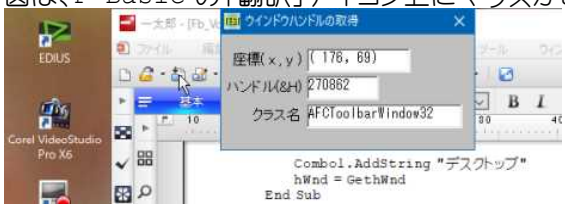

```
'=====================================
'= 親ウィンドウを変更する
'=    (SetParent4.bas)
'=====================================
#include "Windows.bi"

' 指定された子ウィンドウの親ウィンドウを変更
Declare Function Api_SetParent& Lib "user32" Alias "SetParent" (ByVal hWndChild&, ByVal
hWndNewParent&)

Var Shared Combo1 As Object
Var Shared Edit1 As Object

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

Var Shared hWnd As Long

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Combo1.AddString "自フォーム"
```

```
    Combo1.AddString "デスクトップ"
    hWnd = GethWnd
End Sub

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

    '親ウィンドウを設定
    If Combo1.GetCursel = 0 Then
        Ret = Api_SetParent(Edit1.GethWnd, hWnd)
    Else
        Ret = Api_SetParent(Edit1.GethWnd, 0)
    End If
End Sub

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

カーソル位置のウインドウハンドル・クラス名を取得

カーソル位置のウインドウハンドル・クラス名を取得します。
GetCursorPos カーソルの現在のスクリーン座標の取得
WindowFromPoint 指定の座標位置にあるウィンドウハンドルを取得
GetClassName ウィンドウのクラス名を取得

図は、F-Basic の「翻訳」アイコン上にマウスがある状態を示しています。




```
'=======================================
'= カーソル位置のウインドウハンドル・クラス名を取得
'=    (WindowFromPoint.bas)
'=======================================
#include "Windows.bi"

Type POINTAPI
    x   As Long
    y   As Long
End Type

' カーソルの現在のスクリーン座標の取得
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&)

' ウィンドウのクラス名を取得する関数の宣言
Declare Function Api_GetClassName& Lib "user32" Alias "GetClassNameA" (ByVal hWnd&,
ByVal lpClassName$, ByVal nMaxCount&)
```

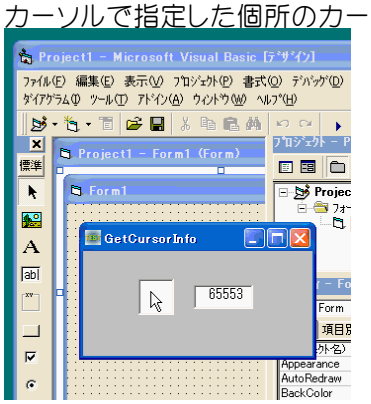
```
Var Shared Timer1 As Object
Var Shared Text(5) As Object

Timer1.Attach GetDlgItem("Timer1")
For i = 0 To 5
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1)))
    Text(i).SetFontSize 14
Next

'=====================================
'=
'=====================================
Declare Sub Mainform_Start edecl ()
Sub Mainform_Start()
    Timer1.SetInterval 50
    Timer1.Enable -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var WinPoint As POINTAPI
    Var hWnd As Long
    Var ClsNameBuf As String * 128
    Var ClsNameLen As Long
    Var Ret As Long

    'カーソル位置のスクリーン座標を取得
    Ret = Api_GetCursorPos(WinPoint)

    '座標を表示
    Text(3).SetWindowText "(" & Str$(WinPoint.x) & "," & Str$(WinPoint.y) & ")"

    '座標を含むウィンドウのハンドルを取得
    hWnd = Api_WindowFromPoint(WinPoint.x, WinPoint.y)

    'ウィンドウのハンドルを取得できたときは
    If hWnd <> 0 Then

        'ウィンドウのハンドルを表示
        Text(4).SetWindowText hex$(hWnd)

        'ウィンドウのクラス名を取得
        ClsNameLen = Api_GetClassName(hWnd, ClsNameBuf, Len(ClsNameBuf))

        'ウィンドウのクラス名を表示
        Text(5).SetWindowText Left$(ClsNameBuf, InStr(ClsNameBuf, Chr$(0)) - 1)
    End If
End Sub

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

カーソル情報を取得

指定した個所のカーソルに関する情報を取得します。
GetCursorInfo グローバルカーソルの情報を取得
DrawIcon アイコンを描画
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

カーソルで指定した個所のカーソルおよびハンドルを表示させています。




```
'=====================================
'= カーソル情報を取得
'=    (GetCursorInfo.bas)
'=====================================
#include "Windows.bi"

Type POINTAPI
    x    As Long
    y    As Long
End Type

Type CURSORINFO
    cbSize              As Long
    flags               As Long
    hCursor             As Long
    ptScreenPos         As POINTAPI
End Type

' グローバルカーソルに関する情報を取得
Declare Function Api_GetCursorInfo& Lib "user32" Alias "GetCursorInfo" (pci As
CURSORINFO)

' アイコンを描画
Declare Function Api_DrawIcon& Lib "user32" Alias "DrawIcon" (ByVal hDC&, ByVal x&, ByVal
y&, ByVal exhIcon&)

' 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 Text1 As Object
Var Shared Timer1 As Object

Picture1.Attach GetDlgItem("Picture1")
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Timer1.Attach GetDlgItem("Timer1")

Var Shared hDC As Long

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    hDC = Api_GetDC(Picture1.GethWnd)
    Timer1.SetInterval 10
    Timer1.Enable -1
End Sub
```

```
'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var ci As CURSORINFO
    Var Ret As Long

    ci.cbSize = Len(ci)

    Ret = Api_GetCursorInfo(ci)

    Text1.SetWindowText Str$(ci.hCursor)

    Picture1.Cls
    Ret = Api_DrawIcon(hDC, 0, 0, ci.hCursor)
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
```

カーソルに影をつける

カーソルに影をつけたり、戻したりします。
SystemParametersInfo システム全体に関するパラメータを取得・設定
SPI_SETCURSORSHADOW(&H101B) カーソルに影をつける
SPIF_SENDWININICHANGE(&H2) 全てのアプリケーションに通知して更新する
SPIF_UPDATEINIFILE(&H1) ユーザープロファイルの更新を指定する

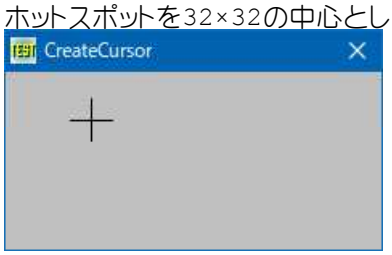

```
'=====================================
'= カーソルに影をつける
'=    (SetCursorShadow.bas)
'=====================================
#include "Windows.bi"

' 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_SETCURSORSHADOW &H101B          'カーソルに影をつける
#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_UPDATEINIFILE &H1              'ユーザープロファイルの更新を指定する
```

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

    'カーソルに影をつけるか設定
    Ret = Api_SystemParametersInfo(SPI_SETCURSORSHADOW, 0, ByVal Index,
SPIF_UPDATEINIFILE Or SPIF_SENDWININICHANGE)
End Sub

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

カーソルを作成

新しいカーソルを作成します。
CreateCursor 指定されたサイズ、ビットパターン及びホットスポットを持つマウスカーソルの作成
DestroyCursor カーソルを破棄する
SetCursor マウスカーソルの形状を設定
Sleep カレントスレッドの実行を指定の時間だけ中断

ホットスポットを32×32の中心とした十字のカーソルを作成し、10秒後にデフォルトカーソルに戻しています。

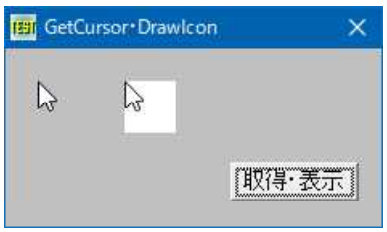



```
'=====================================
'= カーソルを作成
'=    (CreateCursor.bas)
'=====================================
#include "Windows.bi"

' 指定されたサイズ、ビットパターン及びホットスポットを持つマウスカーソルの作成
Declare Function Api_CreateCursor& Lib "user32" Alias "CreateCursor" (ByVal hInstance&,
ByVal nXhotspot&, ByVal nYhotspot&, ByVal nWidth&, ByVal nHeight&, lpANDbitPlane As Any,
lpXORbitPlane As Any)

' カーソルを破棄する
Declare Function Api_DestroyCursor& Lib "user32" Alias "DestroyCursor" (ByVal hCursor&)

' マウスカーソルの形状を設定
Declare Function Api_SetCursor& Lib "user32" Alias "SetCursor" (ByVal hCursor&)

' 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&)
```

```
'====================================
'=
'====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var hNewCursor    As Long               '新たなカーソル
    Var hOldCursor    As Long               'デフォルトカーソル
    Var andBuffer     As String             'カーソルをドットで描画
    Var xorBuffer     As String             'マスク
    Var andBits(127)  As Byte               'ビット(and)
    Var xorBits(127)  As Byte               'ビット(xor)
    Var readData      As String             '読込データ
    Var c             As Integer
    Var Ret           As Long

    'データ読み込み
    For c = 0 To 7
        Read readData
        andBuffer = andBuffer & readData
    Next

    For c = 0 To 7
        Read readData
        xorBuffer = xorBuffer & readData
    Next

    '配列に代入
    For c = 0 To 127
        andBits(c) = Val("&H" & Mid$(andBuffer, 2 * c + 1, 2))
        xorBits(c) = Val("&H" & Mid$(xorBuffer, 2 * c + 1, 2))
    Next c

    'hotspotを中心とした十字のカーソルを作成
    hNewCursor = Api_CreateCursor(GethInst, 16, 16, 32, 32, andBits(0), xorBits(0))

    '新たなカーソルに変更
    hOldCursor = Api_SetCursor(hNewCursor) 'カーソル変更

    Api_Sleep 10000                        '10秒待つ

    Ret = Api_SetCursor(hOldCursor)        'デフォルトカーソルに変更

    Ret = Api_DestroyCursor(hNewCursor)    'カーソルを破棄

    data "FFFF7FFFFFFF7FFFFFFF7FFFFFFF7FFF"
    data "FFFF7FFFFFFF7FFFFFFF7FFFFFFF7FFF"
    data "FFFF7FFFFFFF7FFFFFFF7FFFFFFF7FFF"
    data "FFFF7FFFFFFF7FFFFFFF7FFFFFFF7FFF"
    data "00000000FFFF7FFFFFFF7FFFFFFF7FFF"
    data "FFFF7FFFFFFF7FFFFFFF7FFFFFFF7FFF"
    data "FFFF7FFFFFFF7FFFFFFF7FFFFFFF7FFF"
    data "FFFF7FFFFFFF7FFFFFFF7FFFFFFF7FFF"

    data "00000000000000000000000000000000"
    data "00000000000000000000000000000000"
    data "00000000000000000000000000000000"
    data "00000000000000000000000000000000"
    data "00000000000000000000000000000000"
    data "00000000000000000000000000000000"
    data "00000000000000000000000000000000"
    data "00000000000000000000000000000000"
End Sub

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

カーソルを取得しデバイスコンテキストにアイコン描画

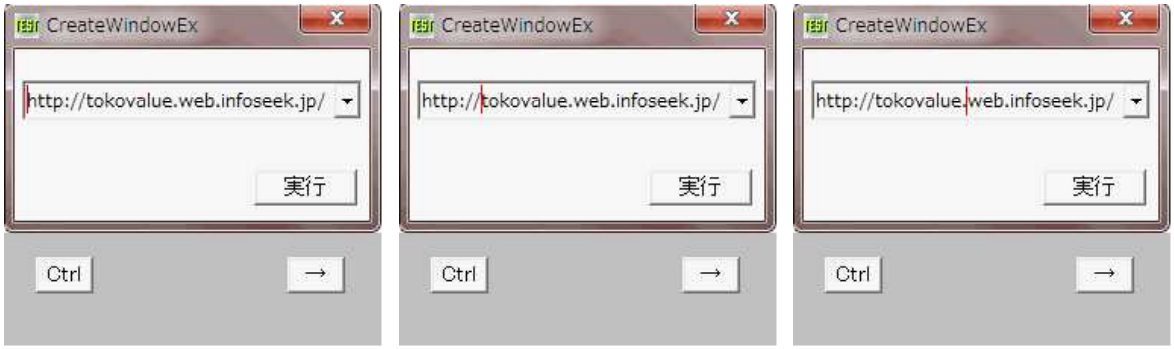
現在のカーソルを取得し、フォームとピクチャボックスに描画しています。
GetCursor　カーソルのハンドルを取得
DrawIcon デバイスコンテキストにアイコンを描画




```
'=====================================
'= カーソルの取得と描画
'=    (DrawIcon.bas)
'=====================================
#include "Windows.bi"

' カーソルのハンドルを取得
Declare Function Api_GetCursor& Lib "user32" Alias "GetCursor" ()

' デバイスコンテキストにアイコンを描画
Declare Function Api_DrawIcon& Lib "user32" Alias "DrawIcon" (ByVal hDC&, ByVal x&, ByVal
y&, 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&)

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

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var fhDC As Long
    Var phDC As Long
    Var cWnd As Long
    Var Ret As Long

    fhDC = Api_GetDC(GethWnd)                'フォームのデバイスコンテキスト
    phDC = Api_GetDC(Picture1.GethWnd)       'ピクチャボックスのデバイスコンテキスト

    cWnd = Api_GetCursor()                   'カーソルのハンドル

    Ret = Api_DrawIcon(fhDC, 20, 20, cWnd)   'フォームに描画
    Ret = Api_DrawIcon(phDC, 0, 0, cWnd)     'ピクチャボックスに描画

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

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

カーソルを単語ごとに移動させる

拡張コンボボックスをコードで作成、URLを表示させ[Ctrl]+[→]でカーソルを単語ごとに移動させます。
GetSysColor システムのCOLOR取得
CreateWindowEX 新しいウインドウ(コントロール)を作成
DestroyWindow ウインドウ(コントロール)の解放
SendMessage ウィンドウにメッセージを送信



※カーソルの位置を赤で示しています。

```
'=======================================
'= カーソルを単語ごとに移動させる
'=    (CreateWindow.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&)

#define COLOR_BTNFACE 15                    '3Dオブジェクトの表面色

Type tagINITCOMMONCONTROLSEX
    dwSize      As Long
    dwICC       As Long
End Type

Type tagCOMBOBOXEXITEM
    mask                As Long
    iItem               As Long
    pszText             As Long
    cchTextMax          As Long
    iImage              As Long
    iSelecteVarage      As Long
    iOverlay            As Long
    iIndent             As Long
    lParam              As Long
End Type

' 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 Function Api_InitCommonControlsEx& Lib "comctl32" Alias "InitCommonControlsEx"
(lpInitCtrls As tagINITCOMMONCONTROLSEX)

' 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)

' CreateWindowExの解放
Declare Function Api_DestroyWindow& Lib "user32" Alias "DestroyWindow" (ByVal hWnd&)

#define CB_SETCURSEL &H14E                  'コンボボックスのリストボックス内の文字列を選択する
#define CBEIF_TEXT &H1                      '
#define CBEM_INSERTITEM &H401               '
```

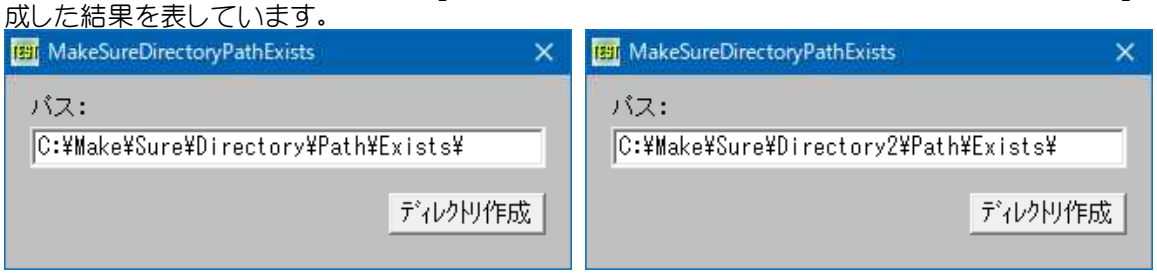
```
#define CBEM_SETEXTENDEDSTYLE &H40E         '
#define CBES_EX_PATHWORDBREAKPROC &H4       '単語ごとにカーソル移動する拡張スタイルを指定
#define CBS_DROPDOWN &H2                    'CBS_SIMPLEで､リストはドロップダウンアイコンで表示
#define CBS_SIMPLE &H1                      '単純なコンボボックス
#define ICC_USEREX_CLASSES &H200            '拡張コンボボックス
#define WC_COMBOBOXEX "ComboBoxEx32"        '拡張コンボボックス
#define WS_CHILD &H40000000                 '親ウインドウを持つコントロールを作成する
#define WS_VISIBLE &H10000000               '可視状態のウィンドウを作成する

Var Shared Button1 As Object

Button1.Attach GetDlgItem("Button1")

Var Shared hComboEx As Long

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var rgbColor As Long
    Var icce As tagINITCOMMONCONTROLSEX
    Var cei As tagCOMBOBOXEXITEM
    Var ccis As Long
    Var WinStyle As Long
    Var InsItemIdx As Long
    Var InsItemStr As String
    Var InsIndex 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

    'コモンコントロールのクラスを指定
    icce.dwSize = Len(icce)
    icce.dwICC = ICC_USEREX_CLASSES

    'コモンコントロールのクラスを登録
    ccis = Api_InitCommonControlsEx(icce)

    If ccis <> 0 Then

        'コンボボックス作成
        hComboEx = Api_CreateWindowEx(0, WC_COMBOBOXEX, "コンボボックスEx", WS_CHILD Or
WS_VISIBLE Or CBS_DROPDOWN, 5, 20, 225, 100, GethWnd, 0, 0, 0)
    End If

    'ComboboxExの末尾に項目追加する指定
    InsItemIdx = -1

    '追加する文字列を指定
    InsItemStr = "http://tokovalue.web.infoseek.jp/"

    '追加する項目情報を構造体に設定
    cei.iItem = InsItemIdx
    cei.mask = CBEIF_TEXT
    cei.pszText = StrAdr(InsItemStr)
    cei.cchTextMax = Len(InsItemStr)

    'ComboboxExの初期値を設定
    InsIndex = Api_SendMessage(hComboEx, CBEM_INSERTITEM, 0, cei)
```

```
    'ComboboxExの末尾に項目追加する指定
    InsItemIdx = (-1)

    '追加する文字列を指定
    InsItemStr = "http://tokovalue.web.infoseek.jp/"

    '追加する項目情報を構造体に設定
    cei.iItem = InsItemIdx
    cei.mask = CBEIF_TEXT
    cei.pszText = StrAdr(InsItemStr)
    cei.cchTextMax = Len(InsItemStr)

    'ComboboxExインデックスを変更
    Ret = Api_SendMessage(hComboEx, CB_SETCURSEL, InsIndex, ByVal CLng(0))
End Sub

'=======================================
'=  [Ctrl] + →
'=======================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var ExStyles As Long
    Var Ret As Long

    'カーソルが単語ごとに移動する拡張スタイルを指定
    ExStyle = CBES_EX_PATHWORDBREAKPROC

    Ret = Api_SendMessage(hComboEx, CBEM_SETEXTENDEDSTYLE, 0, ByVal ExStyle)
End Sub

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

    Ret = Api_DestroyWindow(hComboEx)
End Sub

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

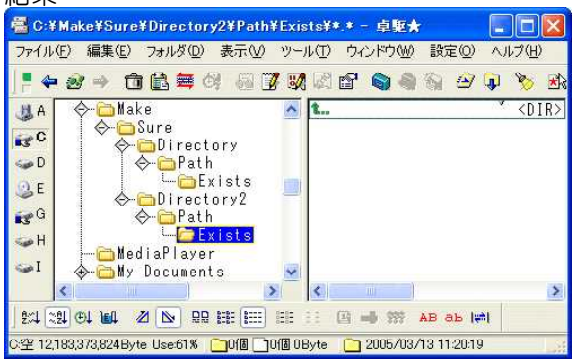
階層化されたフォルダを一気に作成(Ⅰ)

MakeSureDirectoryPathExists 階層化されたフォルダを一気に作成します。

最初に"C:¥Make¥Sure¥Directory¥Exists¥"を作成し、次に"C:¥Make¥Sure¥Directory2¥Exists¥"を作成した結果を表しています。



結果

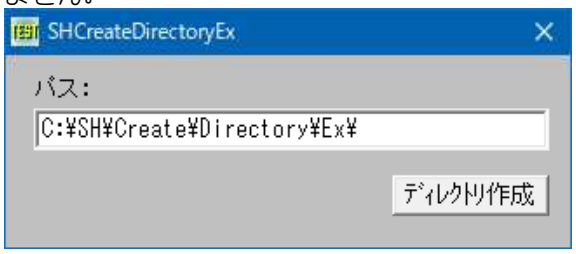



```
'========================================
'= 階層化されたフォルダを一気に作成(Ⅰ)
'=    (MakeSureDirectoryPathExists.bas)
'========================================
#include "Windows.bi"

' 階層化されたフォルダを一気に作成
Declare Function Api_MakeSureDirectoryPathExists& Lib "imagehlp" Alias
"MakeSureDirectoryPathExists" (ByVal lpPath$)

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 DirPath As String

'========================================
'=
'========================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    DirPath = "C:¥Make¥Sure¥Directory¥Path¥Exists¥"
    Edit1.SetWindowText DirPath
End Sub

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

    DirPath = GetDlgItemText("Edit1")

    Ret = Api_MakeSureDirectoryPathExists(DirPath)
End Sub

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

階層化されたフォルダを一気に作成（II）

SHCreateDirectoryEx 階層化されたフォルダを一気に作成

MakeSureDirectoryPathExistsと機能は同じですが、Windows2000以降対応。Windows9x/Meは対応していません。



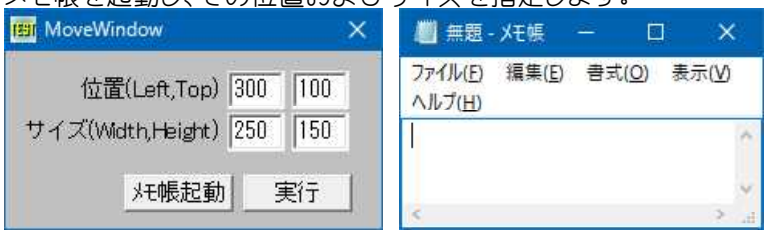

```
'=======================================
'= 階層化されたフォルダを一気に作成（II）
'= Windows2000以降、Windows9x/Meはサポート外
'=    (SHCreateDirectoryEx.bas)
'=======================================
#include "Windows.bi"

' 階層化されたフォルダを一気に作成
Declare Function Api_SHCreateDirectoryEx& Lib "shell32" Alias "SHCreateDirectoryExA"
(ByVal hWnd&, ByVal pszPath$, ByVal psa As Any)

Var Shared Edit1 As Object
Edit1.Attach GetDlgItem("Edit1")

Var Shared DirPath As String

'=======================================
'=
'=======================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    DirPath = "C:¥SH¥Create¥Directory¥Ex¥"
    Edit1.SetWindowText DirPath
End Sub

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

    DirPath = GetDlgItemText("Edit1")

    Ret = Api_SHCreateDirectoryEx(GethWnd, DirPath, ByVal 0)
End Sub

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

外部アプリケーションの位置・サイズを変更

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
MoveWindow 指定されたウィンドウの位置およびサイズを変更

ShellExecute 拡張子に関連付けられたプログラムを実行
SendMessage ウィンドウにメッセージを送信

メモ帳を起動し、その位置およびサイズを指定します。




```
'=====================================
'= 外部アプリケーションの位置・サイズを変更
'=    (MoveWindow.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&)

' 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&)

' 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 SW_SHOWNORMAL 1                    '起動時に通常のウィンドウとして表示
#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

Var Shared Edit(3) As Object
Var Shared Text(1) As Object
Var Shared Button(1) As Object

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

Var Shared hWnd As Long

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    ShowWindow -1
    Cls
End Sub

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

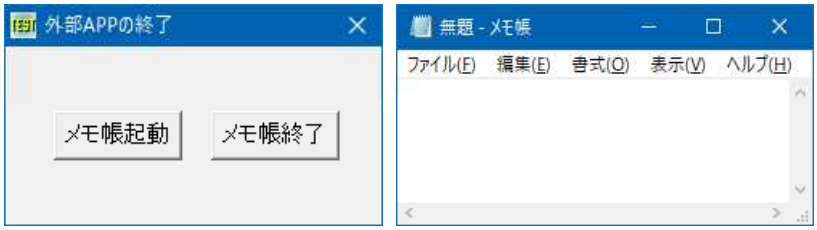
```
    'コマンドボタンを無効に設定
    Button(0).EnableWindow 0

    'メモ帳を起動
    Ret = Api_ShellExecute(GethWnd, ByVal 0, "notepad.exe", ByVal 0, ByVal 0,
SW_SHOWNORMAL)

End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var ClassName As String
    Var mem(3) 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"
    hWnd = Api_FindWindow(ClassName, ByVal 0)

    For i = 0 To 3
       mem(i) = Val(Edit(i).GetWindowText)
    Next

    'ウィンドウハンドルを取得できたとき
    If hWnd <> 0 Then

        'ウィンドウの位置と寸法を設定
        Ret = Api_MoveWindow(hWnd, mem(0), mem(1), mem(2), mem(3), True)
    End If
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_SendMessage(hWnd, WM_CLOSE, 0, 0)
        End
    End If
End Sub

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

外部アプリケーションの終了

電卓を起動しBUTTON2クリック時あらかじめ解っているクラス名から、アプリケーションのハンドルを取得、WM_CLOSE を SendMessageして終了させています。

FindWindow ウインドウハンドル取得
SendMessage 指定ウィンドウにメッセージを送る

例では、フォーム背景色をVisualBasic風の色に塗り潰しています。
マニフェストを作成し、WindowsXPスタイルで…




```
'=====================================
'= 外部アプリケーションの終了
'=    (FindWindow.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&)

' 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
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 COLOR_BTNFACE 15                    '3Dオブジェクトの表面色
#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

'=====================================
'=
'=====================================
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()
    Shell "Notepad.exe"
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var lpClassName$ As String
    Var lpCaption$ As String
    Var hWnd As Long
```

```
    Var Res As Long

    lpClassName$ = "SciCalc"
   'lpClassName$ = "CalcFrame"  'Windows 7
    lpCaption$ = "無題 - メモ帳"

    'メモ帳ウインドウのハンドルを取得
    hWnd = Api_FindWindow(lpClassName$, lpCaption$)

    'メモ帳を終了させる
    Res = Api_SendMessage(hWnd, WM_CLOSE, 0, 0)
End Sub

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

外部アプリケーションの操作

例では、メモ帳を起動し、PictureBox内のマウス位置に応じたサイズ及び位置を設定しています。
SetWindowPos ウィンドウのサイズ、位置、及びZオーダーを設定
ShowWindow 指定されたウィンドウの表示状態を設定
GetWindowRect ウィンドウの座標をスクリーン座標系で取得
FindWindow クラス名、またはキャプションを与えてウィンドウのハンドルを取得
SendMessage ウィンドウにメッセージを送信

0.125および0.25の数値は、Gifファイルを作成する上であまり大きくならないようにするだけのものです。

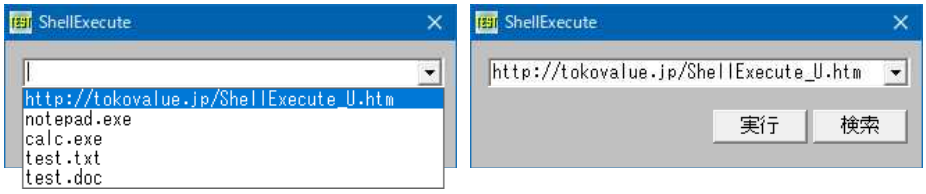

```
'=====================================
'= 外部アプリケーションの操作(II)
'=    (GetWindowRect.bas)
'=====================================
#include "Windows.bi"

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&)

' 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&)
```

```
' 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)

' 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 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 HWND_TOPMOST (-1)                 'ウィンドウを常に最前面に配置
#define SW_MAXIMIZE 3                     '最大化
#define SW_MINIMIZE 6                     '指定のウィンドウをアイコン化しタスクリスト内のトップレベル
                                          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 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

Var Shared Picture1 As Object

Picture1.Attach GetDlgItem("Picture1")

Var Shared hWnd As Long
Var Shared pw As Long
Var Shared ph As Long

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

    pw = Picture1.GetWidth
    ph = Picture1.GetHeight

    Picture1.Symbol(15, 70), "PictureBox内でマウスを移動", 1, 1

    'メモ帳を起動し、そのハンドルを取得
    Shell "notepad.exe", , 5
    Wait 50
    hWnd = Api_FindWindow(ByVal 0, "無題 - メモ帳")

    '最前面へ
    Ret = Api_SetWindowPos(hWnd, HWND_TOPMOST, 0, 0, 0, 0, SWP_NOMOVE Or SWP_NOSIZE)
End Sub

'=====================================
'=
'=====================================
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 rct As RECT
    Var pX As Long
    Var pY As Long
    Var Ret As Long

    pX = x * 1280 / pw
    pY = y * 1024 / ph
    Ret = Api_SetWindowPos(hWnd, 0, pX * 0.25, pY * 0.25, pX * 0.25 + pX * 0.125, pY * 0.25 +
pY * 0.125, 0)

    Ret = Api_GetWindowRect(hWnd, rct)
```

```
End Sub

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

    Ret = Api_ShowWindow(hWnd, SW_MINIMIZE)
End Sub

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

    Ret = Api_ShowWindow(hWnd, SW_RESTORE)
End Sub

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

    Ret = Api_SendMessage(hWnd, WM_CLOSE, 0, 0)
End Sub

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

外部プロセスの起動(Ⅰ)

外部プロセスを起動します。F-BASICのShellExecuteと同じです。
実行ファイルの起動、DOSプロンプトを起動、ドキュメントファイルのオープン、ウェブブラウザの起動を実行します。
ShellExecute 拡張子に関連付けられたプログラムを実行



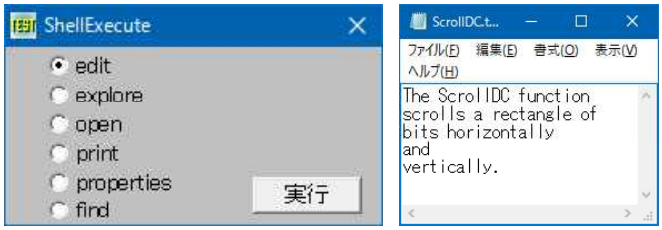

```
'=====================================
'= 外部プロセスの起動
'=    (F-BASICのSHELLEXECUTEと同じ)
'=    (ShellExecute.bas)
'=====================================
#include "Windows.bi"

' 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&)
```

```
#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_MAXIMIZE 3                       '最大化
#define SW_MINIMIZE 6                       '指定のウィンドウをアイコン化しタスクリスト内のトップレベル
                                            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 SW_SHOW 5                           'ウィンドウをアクティブ化し現在の位置とサイズで表示
#define SW_SHOWDEFAULT 10                   '
#define SW_SHOWMAXIMIZED 3                  'ウィンドウをアクティブ化し最大表示
#define SW_SHOWMINIMIZED 2                  'ウィンドウをアクティブ化しアイコン化
#define SW_SHOWMINNOACTIVE 7                'ウィンドウをアイコン化する。現在アクティブなウィンドウは
                                            アクティブなままにする
#define SW_SHOWNA 8                         'ウィンドウを表示する現在アクティブなウィンドウはアクティ
                                            ブなままに
#define SW_SHOWNOACTIVATE 4                 '以前に表示された位置とサイズで表示する。現在アクティ
                                            ブなウィンドウはアクティブなままにする。
#define SW_SHOWNORMAL 1                     '起動時に通常のウィンドウとして表示

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()
    Combo1.AddString "http://tokovalue.jp/ShellExecute_U.htm"
    Combo1.AddString "notepad.exe"
    Combo1.AddString "calc.exe"
    Combo1.AddString "test.txt"
    Combo1.AddString "test.doc"
End Sub

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

    FileName = Combo1.GetText(Combo1.GetCursel)

    'lpOperation$にvbNullStringを指定した場合Openと同じ
    Ret = Api_ShellExecute(GethWnd, ByVal 0, FileName, ByVal 0, ByVal 0, SW_SHOWNORMAL)
End Sub

'=====================================
'= ファイルやフォルダの検索ダイアグラム呼び出し
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Path As String
    Var Ret As Long

    Path = "C:¥Windows"
    Ret = Api_ShellExecute(GethWnd, "Find", Path, ByVal 0, ByVal 0, SW_SHOWNORMAL)
End Sub

'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
```

```
Stop
End
```

外部プロセスの起動(II)

ShellExecute 拡張子に関連付けられたプログラムを実行




```
'=======================================
'= 外部プロセスの起動(II)
'=    (ShellExecute4.bas)
'=======================================
#include "Windows.bi"

' 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&)

#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_MAXIMIZE 3                      '最大化
#define SW_MINIMIZE 6                      '指定のウィンドウをアイコン化しタスクリスト内のトップレベル
                                           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 SW_SHOW 5                          'ウィンドウをアクティブ化し現在の位置とサイズで表示
#define SW_SHOWDEFAULT 10                  '
#define SW_SHOWMAXIMIZED 3                 'ウィンドウをアクティブ化し最大表示
#define SW_SHOWMINIMIZED 2                 'ウィンドウをアクティブ化しアイコン化
#define SW_SHOWMINNOACTIVE 7               'ウィンドウをアイコン化する。現在アクティブなウィンドウは
                                           アクティブなままにする
#define SW_SHOWNA 8                        'ウィンドウを表示する現在アクティブなウィンドウはアクティ
                                           ブなままにする
#define SW_SHOWNOACTIVATE 4                '以前に表示された位置とサイズで表示する。現在アクティ
                                           ブなウィンドウはアクティブなままにする。
#define SW_SHOWNORMAL 1                    '起動時に通常のウィンドウとして表示

#define ERROR_FILE_NOT_FOUND 2             'ファイルが見つからない
#define ERROR_PATH_NOT_FOUND 3             'パスが見つからない
#define ERROR_BAD_FORMAT 11                '不正な形式の実行ファイル
#define SE_ERR_ACCESSDENIED 5              'OSが指定したファイルへのアクセスを拒否
#define SE_ERR_ASSOCINCOMPLETE 27          'ファイルに関連づけられた物が不完全かまたは無効
#define SE_ERR_DDEBUSY 30                  '他のDDEプロセスが通信中だったので、DDE通信が完了で
                                           きなかった
#define SE_ERR_DDEFAIL 29                  'DDE通信が失敗
#define SE_ERR_DDETIMEOUT 28               'DDE通信でタイムアウトが発生
#define SE_ERR_DLLNOTFOUND 32              '指定したDLLファイルが見つからなかった
#define SE_ERR_FNF 2                       'ファイルが見つからなかった
#define SE_ERR_NOASSOC 31                  '指定したファイルに関連づけられたアプリケーションが見つ
                                           からなかった
#define SE_ERR_OOM 8                       '処理を完了するのに十分なメモリがなかった
#define SE_ERR_PNF 3                       '指定したパスが見つからなかった
#define SE_ERR_SHARE 26                    '共有違反が発生
Var Shared Button1 As Object
Var SHared Radio(5) 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

Var Shared Operation(5) As String

'=====================================
'=
'=====================================
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()
    Operation(0) = "edit"
    Operation(1) = "explore"
    Operation(2) = "open"
    Operation(3) = "print"
    Operation(4) = "properties"
    Operation(5) = "find"
End Sub

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

    If Index = 0 Or Index = 3 Then
        Ret = Api_ShellExecute(GethWnd, Operation(Index), "scrolldc.txt", ByVal 0, ByVal
0, SW_SHOWNORMAL)
    Else
        Ret = Api_ShellExecute(GethWnd, Operation(Index), "c:¥fbasicv63", ByVal 0, ByVal
0, SW_SHOWNORMAL)
    End if

    If Ret <= 32 Then
        A% = MessageBox("Err", Str$(Ret), 0, 2)
    End If
End Sub

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

拡大鏡(Ⅰ)

虫眼鏡を作成します。
SendMessage メッセージを送る
SetWindowPos ウィンドウの状態を設定
ReleaseCapture マウスキャプチャの解放
GetDesktopWindow デスクトップのウィンドウハンドル取得
StretchBlt 画像の拡大転送
GetDC ウィンドウハンドルからhDC取得
ReleaseDC hDCを解放
GetCursorPos マウスカーソル位置の取得
CreatePolygonRgn ポリゴン作成
CreateEllipticRgn 円形・楕円形の領域設定

CreateRectRgn 矩形領域の設定
CreateRoundRectRgn 角を丸めた領域の設定
CombineRgn リージョンを結合し新しいリージョンに設定
SetWindowRgn 指定領域をウィンドウ領域に設定

メインフォーム(200×200)にピクチュアボックス(200×200)を重ね合わせます。
ルーペは8～12角形を選んでください(例では12角形)。角数が多いと円形に近くなり、少ない場合は枠より円のほうが大きくなります。
ルーペ倍率は初期で2倍に設定しています。カーソルの部分を拡大します。
タイトルバーのないフォームの移動のリストと併用して、枠部分をドラッグすると移動できるようにしています。
フォームをワンクリックするたびに20%拡大し、4倍を超えると等倍に戻しています。
フォームをダブルクリックで終了させています。

```
'=====================================
'= 拡大鏡(虫眼鏡原型:MainForm12角形)
'= MainForm、Picture1は同サイズ重ね合わせ
'=    (loupe.bas)
'=====================================
#include "Windows.bi"

'点を示すタイプ
Type POINTAPI
    x    As Long
    y    As Long
End Type

' 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&)

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

' 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" ()

' 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&)

' 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)
```

```
' 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&)

' 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&)

' 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&)

' 角の丸い長方形のリージョンを作成
Declare Function Api_CreateRoundRectRgn& Lib "gdi32" Alias "CreateRoundRectRgn" (ByVal
nLeftRect&, ByVal nTopRect&, ByVal nRightRect&, ByVal nBottomRect&, ByVal
nWidthEllipse&, ByVal nHeightEllipse&)

' 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&)

' 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&)

#define COLORONCOLOR 3
#define HTCAPTION 2                     'タイトルバーをクリックしたことを示す
#define WN_NCLBUTTONDOWN &HA1           '非クライアント領域で左マウスボタンを押す
#define RGN_AND 1                       'リージョン同士のAND結合
#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 NULLREGION 1                    'リージョンは空
#define SIMPLEREGION 2                  'リージョンは単一の長方形
#define ERRORAPI 0                      '
#define INDING 2                        '全域モード(塗りつぶし)
#define RCCOPY &HCC0020                 '転送元長方形を転送先長方形にそのままコピー
#define WND_TOPMOST -1                  '
#define SWP_NOACTIVATE &H10             'ウィンドウをアクティブにしない
#define SWP_SHOWWINDOW &H40             'ウィンドウを表示する
#define vbLeftButton 1                  '左ボタンクリック
#define vbRightButton 2                 '右ボタンクリック

Var Shared DtphDC As Long               'デスクトップデバイスコンテキスト
Var Shared PichDC As Long               'ピクチャボックスデバイスコンテキスト
Var Shared srcXY As Integer

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 i As Integer
    Var Corner As Byte                  '角の数
    Var Rad  As Double                  '角度
    Var Rgn1 As Long                    '外枠
    Var Rgn2 As Long                    '円形
    Var Ret  As Long
```

```
    Corner = 12                                 '外周(8～12)
    Var xy(Corner) As POINTAPI                  'Corner角形の頂点位置

    srcXY = 100

    DtphWnd = Api_GetDesktopWindow()            'デスクトップハンドル取得
    PichDC = Api_GetDC(Picture1.GethWnd)        'デバイスコンテキスト取得
    DtphDC = Api_GetDC(DtphWnd)                 'デバイスコンテキスト取得

    Rad = 3.1415926 / (Corner / 2)              'ひとつ分の角度
    For i = 0 To Corner - 1
        xy(i).x = Sin(Rad * i) * (GetWidth / 2) + (GetWidth / 2)
        xy(i).y = Cos(Rad * i) * (GetHeight / 2) + (GetHeight / 2)
    Next i

    Rgn1 = Api_CreatePolygonRgn(xy(0), Corner, INDING)                  '外枠
    Ret = Api_SetWindowRgn(GethWnd, Rgn1, -1)

    Rgn2 = Api_CreateEllipticRgn(10, 10, GetWidth - 9, GetHeight - 9    '円形
    Ret = Api_SetWindowRgn(Picture1.GethWnd, Rgn2, -1)

    Ret = Api_SetWindowPos(GethWnd, WND_TOPMOST, 0, 0, GetWidth, GetHeight,
SWP_SHOWWINDOW Or SWP_NOACTIVATE)

    ShowWindow -1                                                       '形状設定後表示

    Timer1.SetInterval 10
    Timer1.Enable -1
End Sub

'=====================================
'= カーソル位置のデスクトップ画像を拡大して転送
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var xy As POINTAPI
    Var Ret As Long

    Ret = Api_GetCursorPos(xy)
    Ret = Api_SetStretchBltMode(PichDC, COLORONCOLOR)
    Ret = Api_StretchBlt(PichDC, 0, 0, 200, 200, DtphDC, xy.x - (srcXY / 2), xy.y - (srcXY /
2), srcXY, srcXY, RCCOPY)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_MouseDown edecl (Button As Integer, Shift As Integer, x As Single, y
As Single)
Sub MainForm_MouseDown(Button As Integer, Shift As Integer, x As Single, y As Single)
    Var Ret As Long

    Ret = Api_ReleaseCapture()
    Ret = Api_SendMessage(GethWnd, WN_NCLBUTTONDOWN, HTCAPTION, 0)
End Sub

'=====================================
'= 拡大(+)
'=====================================
Declare Sub Picture1_Click edecl ()
Sub Picture1_Click()
    srcXY = srcXY - 20
    If srcXY < 10 Then srcXY = 200
End Sub

'=====================================
'= 終了(ピクチャダブルクリック)
'=====================================
Declare Sub Picture1_DblClick edecl ()
Sub Picture1_DblClick()
```

```
    Var Ret As Long

    Ret = Api_ReleaseDC(Picture1.GethWnd, PichDC)
    Ret = Api_ReleaseDC(DtphWnd, DtphDC)
    End
End Sub

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

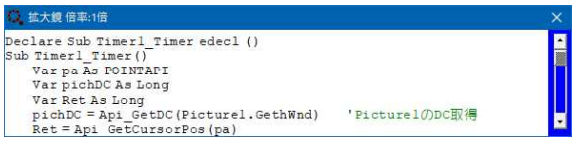
拡大鏡(II)

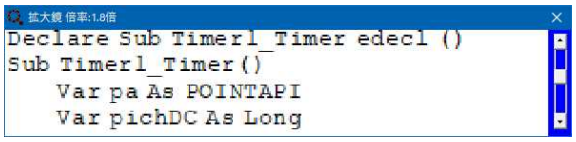
通常フォームの拡大鏡を作成します。
SendMessage ウィンドウにメッセージを送信
SetWindowPos ウィンドウのサイズ、位置、および Z オーダーを設定
ReleaseCapture マウスのキャプチャを解放
GetDesktopWindow Windowsのデスクトップウィンドウを識別
StretchBlt 拡縮をともなうグラフィックデバイス間のイメージを転送
GetDC デバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放
GetCursorPos マウスカーソル(マウスポインタ)の現在の位置に相当するスクリーン座標を取得

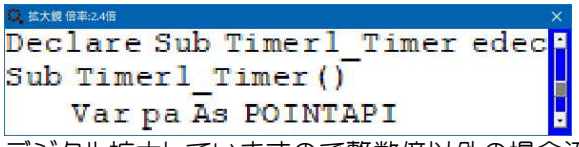
四角い拡大鏡です。少し広い視野を確保します。老眼鏡でも、『,』『.』『;』『:』の判別が困難なのです。^^;)
ついでに、倍率を可変させてみました(1～3倍)。必要の有無は別としてテストですので…
フォームにPictureBox、VScroll、Timerを貼り付けます。アイコン化あり、最大表示なし。
フォームサイズ、PictureBoxサイズはプログラム上で指定しています。あまり大きくすると動作が遅くなります。
SetWindowPosで最前面固定に設定しています。
フォームのプロパティでフレームの種類は境界線にしておきましょう。うっかりフォームサイズを大きくしてしまうとタイヘンです。

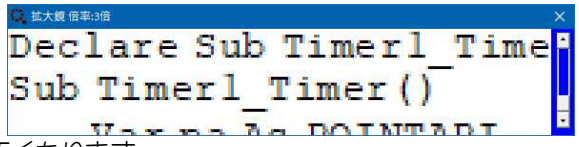
実行時の大きさ(たったこれだけでWindowsXPのアクセサリ→拡大鏡より使いやすいかも••)
カーソル位置(赤で表示)から拡大します。

```
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var pa As POINTAPI
    Var pichDC As Long
    Var Ret As Long
```









デジタル拡大していますので整数倍以外の場合汚くなります。

```
'=====================================
'=  拡大鏡
'=  StretchBlt
'=    (Loupe2.bas)
'=====================================
```

```
#include "Windows.bi"

Type POINTAPI
    X   As Long
    Y   As Long
End Type

' 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&)

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

' 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" ()

' 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&)
' 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)

#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 HWND_TOPMOST (-1)                   'ウィンドウを常に最前面に配置
#define SWP_SHOWWINDOW &H40                 'ウィンドウを表示する
#define SWP_NOSIZE &H1                      'ウィンドウの現在のサイズを保持する
#define SWP_NOMOVE &H2                      'ウィンドウの現在位置を保持する
#define HTCAPTION 2                         'タイトルバーをクリックしたことを示す
#define RCCOPY &HCC0020                     '転送元長方形を転送先長方形にそのままコピー
#define COLORONCOLOR 3                      '取り除く点の情報を保存することなく削除

Var Shared hWnd As Long
Var Shared dtphDC As Long
Var Shared FormWidth As Long
Var Shared FormHeight As Long
Var Shared N As single                      '倍率

Var Shared Picture1 As Object
Var Shared Timer1 As Object
Var Shared VScroll1 As Object

Picture1.Attach GetDlgItem("Picture1")
Timer1.Attach GetDlgItem("Timer1")
VScroll1.Attach GetDlgItem("VScroll1")

'=====================================
'=
'=====================================
declare sub MainForm_Start edecl ()
sub MainForm_Start()
    Var dtphWnd As Long

    FormWidth = 600
    FormHeight = 140
    SetWindowSize FormWidth, FormHeight
```

```
    Picture1.SetWindowSize FormWidth - 30, FormHeight
    Picture1.MoveWindow 0, 0

    VScroll1.SetWindowSize 14, FormHeight - 40
    VScroll1.MoveWindow FormWidth - 24, 4
    VScroll1.SetScrollRange 10, 30
    VScroll1.SetScrollStep 5, 1
    VScroll1.SetScrollPos 20
    N = 2
    SetWindowtext "拡大鏡 倍率:" & Trim$(Str$(N)) & "倍"

    MoveWindow 0, 0
    ShowWindow -1

    dtphWnd = Api_GetDesktopWindow()        'デスクトップハンドル取得
    dtphDC = Api_GetDC(dtphWnd)             'デバイスコンテキスト取得

    Timer1.SetInterval 1
    Timer1.Enable -1
End sub

'=====================================
'= カーソル位置のデスクトップ画像を拡大(2倍)して転送
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var pa As POINTAPI
    Var pichDC As Long
    Var Ret As Long
    pichDC = Api_GetDC(Picture1.GethWnd)    'Picture1のDC取得
    Ret = Api_GetCursorPos(pa)
    Ret = Api_SetWindowPos(GethWnd, HWND_TOPMOST, 0, 0, 0, 0, SWP_SHOWWINDOW Or SWP_NOMOVE
Or SWP_NOSIZE)
    Ret = Api_StretchBlt(pichDC, 0, 0, FormWidth * N, FormHeight * N, dtphDC, pa.X, pa.Y -
10, FormWidth, FormHeight, RCCOPY)
End Sub

'=====================================
'=
'=====================================
Declare Sub VScroll1_Change edecl ()
Sub VScroll1_Change()
    N = VScroll1.GetScrollPos / 10
    SetWindowtext "拡大鏡 倍率:" & Trim$(Str$(N)) & "倍"
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_QueryClose(Cancel%, ByVal Mode%)
Sub MainForm_QueryClose(Cancel%, ByVal Mode%)
    Var Ret As Long

    Ret = Api_ReleaseDC(hWnd, hDC)
    End
End sub
```

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

拡大鏡(III)

虫眼鏡を作成します。 拡大鏡(I)と同じですが、拡大・縮小およびスムーズ拡大(?)、終了を右クリックで選択できるようにしてみました。
SystemParametersInfo システム全体に関するパラメータを取得・設定
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
SendMessage メッセージを送る
SetWindowPos ウィンドウの状態を設定
ReleaseCapture マウスキャプチャの解放
GetDesktopWindow デスクトップのウィンドウハンドル取得
StretchBlt 画像の拡大転送
GetDC ウィンドウハンドルから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
CreatePolygonRgn 多角形のリージョンを作成
CreateEllipticRgn 楕円形のリージョンを作成
CreateRectRgn 長方形のリージョンを作成
CreateRoundRectRgn 角の丸い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
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
AnimateWindow フォームをアニメーション表示

メインフォーム(200×200)にピクチュアボックス(200×200)を重ね合わせます。
ルーペは8～12角形を選んでください(例では12角形)。角数が多いと円形に近くなり、少ない場合は枠より円のほうが大きくなります。
ルーペ倍率は初期で2倍に設定しています。カーソルの部分を拡大します。
ルーペの移動は、青いフレーム部分をドラッグします。
フレーム(青い枠)部を右クリックすると、クリックした位置を左上とするポップアップメニュー(Form2)が表示されます。

拡大縮小	スクロールバーで指定します。拡大縮小率は、その都度表示されます。
拡大縮小範囲	0.8倍 ～ 4倍
通常拡大	StretchBltでの拡大縮小転送です。
スムーズ	SetStretchBltModeを使って綺麗に(?)拡大縮小します。
設定終了	ポップアップメニューを消します。
ルーペの終了	アニメーション終了(あまり効いていません)させています。

通常拡大　　スムーズ拡大

```
'====================================
'= 虫眼鏡　(リージョン:MainForm12角形)
'=    MainForm、Picture1は同サイズ重ね合わせ
'=    (Loupe4.bas)
'====================================
#include "Windows.bi"

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

' 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 RECT, ByVal fWinIni&)

' 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)

' 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&)

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

' 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" ()

' 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&)

' 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&)

' 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&)

' 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&)

' 角の丸い長方形のリージョンを作成
Declare Function Api_CreateRoundRectRgn& Lib "gdi32" Alias "CreateRoundRectRgn" (ByVal
nLeftRect&, ByVal nTopRect&, ByVal nRightRect&, ByVal nBottomRect&, ByVal
nWidthEllipse&, ByVal nHeightEllipse&)
```

```
' 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&)

' 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&)

' フォームをアニメーション表示
Declare Function Api_AnimateWindow& Lib "user32" Alias "AnimateWindow" (ByVal hWnd&,
ByVal dwTime&, ByVal dwFlags&)

#define COLORONCOLOR 3
#define FORM_HIDE &H80000 Or &h10000        '複合操作
#define HALFTONE 4                          'ハーフトーン
#define HTCAPTION 2                         'タイトルバーをクリックしたことを示す
#define INDING 2                            '全域モード(塗りつぶし)
#define MF_STRING &H0                       '文字列
#define RCCOPY &HCC0020                     '転送元長方形を転送先長方形にそのままコピー
#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_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_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 WN_NCLBUTTONDOWN &HA1               '非クライアント領域で左マウスボタンを押す
#define WND_TOPMOST -1                      '
#define SWP_NOACTIVATE &H10                 'ウィンドウをアクティブにしない
#define SWP_SHOWWINDOW &H40                 'ウィンドウを表示する
#define SPI_GETWORKAREA 48

Var Shared dhDC As Long                     'デスクトップデバイスコンテキスト
Var Shared phDC As Long                     'ピクチャボックスデバイスコンテキスト
Var Shared srcXY As Integer
Var Shared hMenu As Long
Var Shared Flag As Long
Var Shared pa As POINTAPI
Var Shared rc As RECT

'---------- MainForm ------------------------------------------------
Var Shared Picture1 As Object
Var Shared Timer1 As Object

Picture1.Attach GetDlgItem("Picture1")
Timer1.Attach GetDlgItem("Timer1")

'---------- Form2 ---------------------------------------------------
Var Shared Form2 As Object
Var Shared Text1 As Object
Var Shared HScroll1 As Object
Var Shared Check1 As Object
Var Shared Button1 As Object
Var Shared Button2 As Object

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var i As Integer
    Var Corner As Byte                      '角の数
    Var Rad  As Double                      '角度
    Var Rgn1 As Long                        '外枠
    Var Rgn2 As Long                        '円形
    Var Ret  As Long
```

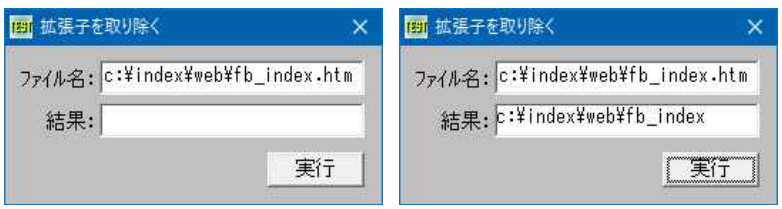
```
    'ワークエリア取得
    Ret = Api_SystemParametersInfo(SPI_GETWORKAREA, 0, rc, 0)

    Flag = 0

    '外周(8～12)
    Corner = 12

    'Corner角形の頂点位置
    Var xy(Corner) As POINTAPI

    srcXY = 100

    'デスクトップハンドル取得
    DtphWnd = Api_GetDesktopWindow()

    'PictureBoxデバイスコンテキスト取得
    phDC = Api_GetDC(Picture1.GethWnd)

    'デスクトップデバイスコンテキスト取得
    dhDC = Api_GetDC(DtphWnd)

    'ひとつ分の角度
    Rad = 3.1415926 / (Corner / 2)

    For i = 0 To Corner - 1
        xy(i).x = Sin(Rad * i) * (GetWidth / 2) + (GetWidth / 2)
        xy(i).y = Cos(Rad * i) * (GetHeight / 2) + (GetHeight / 2)
    Next i

    '外枠
    Rgn1 = Api_CreatePolygonRgn(xy(0), Corner, INDING)
    Ret = Api_SetWindowRgn(GethWnd, Rgn1, -1)

    '円形
    Rgn2 = Api_CreateEllipticRgn(10, 10, GetWidth - 9, GetHeight - 9)
    Ret = Api_SetWindowRgn(Picture1.GethWnd, Rgn2, -1)

    Ret = Api_SetWindowPos(GethWnd, WND_TOPMOST, 0, 0, GetWidth, GetHeight,
SWP_SHOWWINDOW Or SWP_NOACTIVATE)

    '形状設定後表示
    ShowWindow -1

    Timer1.SetInterval 10
    Timer1.Enable -1
End Sub

'=====================================
'= カーソル位置のデスクトップ画像を拡大して転送
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var Ret As Long

    Ret = Api_GetCursorPos(pa)

    If Flag = 0 Then

        'そのまま拡大縮小
        Ret = Api_SetStretchBltMode(phDC, COLORONCOLOR)
    Else

        '綺麗に拡大縮小
        Ret = Api_SetStretchBltMode(phDC, HALFTONE)
    End if

    Ret = Api_StretchBlt(phDC, 0, 0, 200, 200, dhDC, pa.x - (srcXY / 2), pa.y - (srcXY / 2),
srcXY, srcXY, RCCOPY)
End Sub
```

```
'====================================
'=
'====================================
Declare Sub MainForm_MouseDown edecl (Button As Integer, Shift As Integer, x As Single, y
As Single)
Sub MainForm_MouseDown(Button As Integer, Shift As Integer, x As Single, y As Single)
    Var Ret As Long

    Ret = Api_ReleaseCapture()
    Ret = Api_SendMessage(GethWnd, WN_NCLBUTTONDOWN, HTCAPTION, 0)
End Sub

'====================================
'= 設定フォーム
'====================================
Declare Sub MainForm_MouseUp edecl (Button As Integer, Shift As Integer, x As Single, y As
Single)
Sub MainForm_MouseUp(Button As Integer, Shift As Integer, x As Single, y As Single)
    Var Flags As Long
    Var Ret As Long

    If Not(CheckObject(Form2)) Then
        Form2.CreateWindow "Form2", -1
        Text1.Attach Form2.GetDlgItem("Text1") : Text1.SetFontSize 14
        HScroll1.Attach Form2.GetDlgItem("HScroll1")
        Check1.Attach Form2.GetDlgItem("Check1") : Check1.SetFontSize 14
        Button1.Attach Form2.GetDlgItem("Button1") : Button1.SetFontSize 14
        Button2.Attach Form2.GetDlgItem("Button2") : Button2.SetFontSize 14

        Form2.SetTopMostWindow -1

        'マウスカーソルの位置を取得
        Ret = Api_GetCursorPos(pa)

        '設定画面(Form2)が、ワークエリア外に位置する場合の処理
        If pa.x + Form2.GetWidth > rc.Right And pa.y + Form2.GetHeight > rc.Bottom Then

            '縦横共にエリア外の場合
            Form2.MoveWindow rc.Right - 200, rc.Bottom - 130

        Else If pa.x + Form2.GetWidth > rc.Right Then

            '横のみエリア外の場合
            Form2.MoveWindow rc.Right - 200, pa.y

        Else If pa.y + Form2.GetHeight > rc.Bottom Then

            '縦のみエリア外の場合
            Form2.MoveWindow pa.x, rc.Bottom - 130

        Else
            'アリア内の場合
            Form2.MoveWindow pa.x, pa.y
        End If

        HScroll1.SetScrollRange 50, 250
        HScroll1.SetScrollStep 10, 10
        HScroll1.SetScrollPos 300 - srcXY
        Text1.SetWindowText "拡大縮小率:" & Format$(200 / srcXY, "#.#")
        Check1.SetCheck Flag
        Flag = Check1.GetCheck
    End If
End Sub

'====================================
'= スムーズ処理
'====================================
Declare Sub Check1_on edecl ()
Sub Check1_on()
    Flag = Check1.GetCheck
```

```
End Sub

'=====================================
'= 拡大率設定
'=====================================
Declare Sub HScroll1_Change edecl ()
Sub HScroll1_Change()
    srcXY = 300 - HScroll1.GetScrollPos
    Text1.SetWindowText "拡大縮小率:" & Format$(200 / srcXY, "#.#")
End Sub

'=====================================
'= 設定終了
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Form2.DestroyWindow
    Text1.Detach
    HScroll1.Detach
    Check1.Detach
    Button1.Detach
    Button2.Detach
End Sub

'=====================================
'= 拡大鏡の終了
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Form2.DestroyWindow
    Text1.Detach
    HScroll1.Detach
    Check1.Detach
    Button1.Detach
    Button2.Detach

    Ret = Api_ReleaseDC(Picture1.GethWnd, phDC)
    Ret = Api_ReleaseDC(DtphWnd, dhDC)
    Ret = Api_AnimateWindow(GethWnd, 1000, FORM_HIDE)
    End
End Sub

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

拡張子を取り除く

拡張子を取り除きます。
PathRemoveExtension 拡張子を取り除く



```
'=====================================
'= Extension(拡張子を取り除く)
'=    (PathRemoveExtension.bas)
'=====================================
#include "Windows.bi"

' パス文字列から拡張子を取り除く関数
Declare Sub Api_PathRemoveExtension Lib "shlwapi" Alias "PathRemoveExtensionA" (ByVal
pszPath$)

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

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

'=====================================
'=
'=====================================
Declare Sub Button1_On edecl ()
Sub Button1_On()
    Var File As String

    File = Edit1.GetWindowText
    If File = "" Then Exit Sub

    '拡張子を取り除く
    Api_PathRemoveExtension File

    Text(2).SetWindowText Left$(File, InStr(File, Chr$(0)) - 1)
End Sub

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

拡張メタファイルの作成

拡張メタファイルを作成します。通常メタファイルの拡張子は.wmfですが拡張メタファイル(エンハンストメタファイル)は.emfです。
GetDesktopWindow デスクトップウィンドウの識別
GetWindowDC ウィンドウ全体のデバイスコンテキストを取得
CreateEnhMetaFile 拡張メタファイルの新規作成
PlayEnhMetaFile 拡張メタファイルの描画
CloseEnhMetaFile 拡張メタファイル用デバイスコンテキストのクローズ
DeleteEnhMetaFile 拡張メタファイルの削除
GetDeviceCaps デバイス固有の情報を取得
GetClientRect ウィンドウのクライアント領域の座標を取得
GetWindowRect ウィンドウの座標をスクリーン系座標で取得
BitBlt ビットブロック転送
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
GetDC デバイスコンテキスト取得
ReleaseDC デバイスコンテキストの解放

作成されたデスクトップイメージのメタファイルを、ピクチャボックスに縮小表示しています。
保存されたメタファイルTempemf.emfの確認(右)

FBASIC V6.3のサンプルでの再生画例(プログラム下段参照)


```
'=======================================
'= メタファイルの作成
'=    (CreateEnhMetaFile2.bas)
'=======================================
#include "Windows.bi"

#define HORZSIZE 4                          '物理画面の幅(ミリメートル単位)
#define VERTSIZE 6                          '物理画面の高さ(ミリメートル単位)
#define HORZRES 8                           '画面の幅(ピクセル単位)
#define VERTRES 10                          '画面の高さ(ピクセル単位)
#define STRETCH_ANDSCANS 1                  '既存のカラー値とAND演算
#define STRETCH_DELETESCANS 3               'コピー先のピクセルをコピー元のピクセルで置き換え
#define STRETCH_HALFTONE 4                  'コピー先のピクセルの平均カラー値を取得
#define STRETCH_ORSCANS 2                   '既存のカラー値とOR演算
#define SRCCOPY &HCC0020                    'そのまま転送

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

' 拡張メタファイルの新規作成
Declare Function Api_CreateEnhMetaFile& Lib "gdi32" Alias "CreateEnhMetaFileA" (ByVal
hdcRef&, ByVal lpFileName$, ByRef lpRect As RECT, ByVal lpDescription$)

' 拡張メタファイル用デバイスコンテキストのクローズ
Declare Function Api_CloseEnhMetaFile& Lib "gdi32" Alias "CloseEnhMetaFile" (ByVal hDC&)

' 拡張メタファイルの削除
Declare Function Api_DeleteEnhMetaFile& Lib "gdi32" Alias "DeleteEnhMetaFile" (ByVal
hEmf&)

' 拡張メタファイルの描画
Declare Function Api_PlayEnhMetaFile& Lib "gdi32" Alias "PlayEnhMetaFile" (ByVal hDC&,
ByVal hEmf&, ByRef lpRect 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&)

' デバイス固有の情報を取得
Declare Function Api_GetDeviceCaps& Lib "gdi32" Alias "GetDeviceCaps" (ByVal hDC&, ByVal
```

```
nIndex&)

' 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)

' 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)

' 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

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

'=====================================
'= デバイスコンテキストからメタファイル作成
'=====================================
Declare Function DcToEmf2(ByVal hDcIn As Long, inArea As Rect, sOutputFile As String) As
Long
Function DcToEmf2(ByVal hDcIn As Long, inArea As Rect, sOutputFile As String) As Long
    Var rc As Rect
    Var MetaDC As Long
    Var OldMode As Long
    Var hsize As Long
    Var vsize As Long
    Var hres As Long
    Var vres As Long
    Var Ret As Long

    hsize = Api_GetDeviceCaps(hDcIn, HORZSIZE) * 100
    vsize = Api_GetDeviceCaps(hDcIn, VERTSIZE) * 100
    hres = Api_GetDeviceCaps(hDcIn, HORZRES)
    vres = Api_GetDeviceCaps(hDcIn, VERTRES)

    rc.Left = (inArea.Left * hsize) / hres
    rc.Top = (inArea.Top * vsize) / vres
    rc.Right = (inArea.Right * hsize) / hres
    rc.Bottom = (inArea.Bottom * vsize) / vres

    'メタファイルのデバイスコンテキスト
    MetaDC = Api_CreateEnhMetaFile(hDcIn, sOutputFile, rc, ByVal 0)

    If MetaDC Then

        'メタファイルのデバイスコンテキストに伸縮モード設定
        OldMode = Api_SetStretchBltMode(MetaDC, STRETCH_HALFTONE)

        'メタファイルデバイスコンテキストにデスクトップデバイスコンテキストを伸縮コピー
        Ret = Api_BitBlt(MetaDC, 0, 0, (inArea.Right - inArea.Left), (inArea.Bottom -
inArea.Top), hDcIn, inArea.Left, inArea.Top, SRCCOPY)

        Ret = Api_SetStretchBltMode(MetaDC, OldMode)
        DcToEmf2 = Api_CloseEnhMetaFile(MetaDC)
    End If
```

```
End Function

'=====================================
'= クライアントからメタファイルへ
'=====================================
Declare Function WindowClientToEMF(ByVal hwndIn As Long, sOutputFile As String) As Long
Function WindowClientToEMF(ByVal hwndIn As Long, sOutputFile As String) As Long
    Var rc As Rect
    Var hTmpDc As Long
    Var Ret As Long

    hTmpDC = Api_GetDC(hwndIn)

    If hTmpDC <> 0 Then
        If Api_GetClientRect(hwndIn, rc) <> 0 Then
            WindowClientToEMF = DcToEmf2(hTmpDC, rc, sOutputFile)
            Ret = Api_ReleaseDC(hwndIn, hTmpDC)
        End If
    End If
End Function

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
   Button1.SetWindowText "メタファイル作成"
End Sub

'=====================================
'= メタファイル作成・保存・描画
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hEMF As Long
    Var rc As Rect
    Var hDC As Long
    Var Ret As Long

    hEMF = WindowClientToEMF(Api_GetDesktopWindow(), "TempEMF.emf")

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

    Picture1.Cls

    'Picture1のクライアント領域の座標を取得
    Ret = Api_GetClientRect(Picture1.GethWnd, rc)

    'Picture1にメタファイル描画
    Ret = Api_PlayEnhMetaFile(hDC, hEMF, rc)

    '拡張メタファイルの削除
    Ret = Api_DeleteEnhMetaFile(hEMF)

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

'=====================================
'= ピクチャボックスクリア
'=====================================
Declare Sub Picture1_Click edecl ()
Sub Picture1_Click()
    Picture1.Cls
End Sub

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

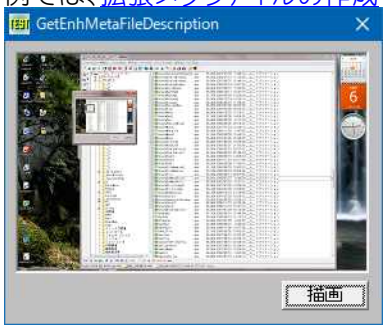
```
While 1
    WaitEvent
Wend
Stop
End
```

```
'=====================================
'=  メタファイルの描画サンプルプログラム
'=  COPYRIGHT FUJITSU LIMITED 1995-1999
'=  FBASICV6.3附属SAMPLEより(ファイル名のみ変更)
'=====================================
#include "Windows.bi"

Var Shared MetaF As Object

MetaFileObject MetaF

Declare Sub MainForm_Start edecl ()
Sub mainForm_Start()

    MetaF.LoadFile "Tempemf.emf"
    PlayMetaFile MetaF
    MetaF.DeleteObject
End Sub

While 1
    WaitEvent
Wend
Stop
End
```

拡張メタファイルの描画

既存の拡張メタファイルを読み込み描画します。
GetEnhMetaFileDescription 拡張メタアプリケーション名の取得
GetEnhMetaFile 拡張メタファイルのオープン
PlayEnhMetaFile 拡張メタファイルの描画
DeleteEnhMetaFile 拡張メタファイルの削除
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

例では、拡張メタファイルの作成で作成されたメタファイルを読み込み描画しています。



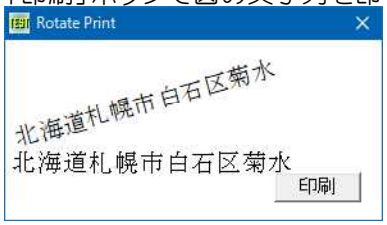

```
'=====================================
'= 拡張メタファイルの描画
'=    (GetEnhMetaFileDescription.bas)
'=====================================
#include "Windows.bi"

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

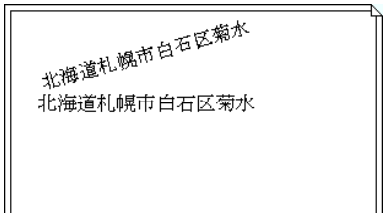
```
End Type

' 拡張メタアプリケーション名の取得
Declare Function Api_GetEnhMetaFileDescription& Lib "gdi32" Alias
"GetEnhMetaFileDescriptionA" (ByVal hEmf&, ByVal DescSize&, ByVal Descrption$)

' 拡張メタファイルのオープン
Declare Function Api_GetEnhMetaFile& Lib "gdi32" Alias "GetEnhMetaFileA" (ByVal
MetaFileName$)

' 拡張メタファイルの描画
Declare Function Api_PlayEnhMetaFile& Lib "gdi32" Alias "PlayEnhMetaFile" (ByVal hDC&,
ByVal hEmf&, ByRef rct As RECT)

' 拡張メタファイルの削除
Declare Function Api_DeleteEnhMetaFile& Lib "gdi32" Alias "DeleteEnhMetaFile" (ByVal
hEMF&)

' 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

Picture1.Attach GetDlgItem("Picture1")
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hDC As Long
    Var rct As RECT
    Var hEmf As Long
    Var Ret As Long

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

    '既存拡張メタファイルオープン
    hEmf = Api_GetEnhMetaFile("F:¥_FB_API_E¥__C¥tempEMF.emf")

    '描画領域を設定
    rct.Left = 0
    rct.Top = 0
    rct.Right = Picture1.GetWidth
    rct.Bottom = Picture1.GetHeight

    '拡張メタファイル描画
    Ret = Api_PlayEnhMetaFile(hDC, hEmf, rct)

    '拡張メタファイルクローズ
    Ret = Api_DeleteEnhMetaFile(hEmf)

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

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

角度のある文字列印刷

指定した角度で文字列を描画し印刷を実行します。
OpenPrinter プリンタオブジェクトをオープン
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、関連づけされた全てのリソースを解放
StartDoc 印刷ジョブの開始
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ページ書き込み終了を通知
EndDoc 印刷ジョブの終了
DeleteDC 指定されたデバイスコンテキストを削除
CreateDC 指定されたデバイスコンテキストを、指定された名前で作成
GetDeviceCaps デバイス固有の情報を取得
CreateFont 指定の属性を持つ論理フォントを作成
TextOut 文字を描画
MulDiv 2つの符号付き32ビット整数を乗算(64ビット)し、その結果を1つの符号付き32ビット整数で除算
GetDC デバイスコンテキストのハンドルを取得
FillRect ブラシで矩形領域を塗りつぶす
SetRect RECT構造体の値を設定
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
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
BitBlt ビットブロック転送

「印刷」ボタンで図の文字列を印字します。プリンタ名は明示的に指定しています。右は印刷された状態






```
'=======================================
'= 角度のある文字列印刷
'=    (RotatePrint.bas)
'=======================================
#include "Windows.bi"

Type DOCINFO
    cbSize              As Long
    lpszDocName         As Long
    lpszOutput          As Long
End Type

Type PRINTER_DEFAULTS
    pDatatype           As Long
    pDevMode            As Long
    DesiredAccess       As Long
End Type

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

' プリンタオブジェクトをオープン
Declare Function Api_OpenPrinter& Lib "winspool.drv" Alias "OpenPrinterA" (ByVal
pPrinterName$, phPrinter&, pDefault As PRINTER_DEFAULTS)

' 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&)
```

```
' 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$, ByVal
lpDevName$, ByVal lpOutput$, ByVal lpInitData&)

' デバイス固有の情報を取得
Declare Function Api_GetDeviceCaps& Lib "gdi32" Alias "GetDeviceCaps" (ByVal hDC&, ByVal
nIndex&)

' 指定の属性を持つ論理フォントを作成
Declare Function Api_CreateFont& Lib "gdi32" Alias "CreateFontA" (ByVal Hei&, ByVal Wid&,
ByVal Esc&, ByVal Ori&, ByVal Wei&, ByVal Ita&, ByVal Und&, ByVal Stk&, ByVal Chr&, ByVal
Out&, ByVal Clp&, ByVal Qua&, ByVal Pit&, ByVal Face$)

' 文字を描画
Declare Function Api_TextOut& Lib "gdi32" Alias "TextOutA" (ByVal hDC&, ByVal nXStart&,
ByVal nYStart&, ByVal lpString$, ByVal cbString&)

' 2つの符号付き32ビット整数を乗算(64ビット)し、その結果を1つの符号付き32ビット整数で除算
Declare Function Api_MulDiv& Lib "Kernel32" Alias "MulDiv" (ByVal nNumber&, ByVal
nNumerator&, ByVal nDenominator&)

' 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&)

' 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&)

' 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&)

' 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&)

'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 PS_SOLID 0
#define PRINTER_ACCESS_ADMINISTER &H4      'プリンタアクセス権の管理者権限を示す定数の宣言
#define PRINTER_ACCESS_USE &H8             'プリンタアクセス権のユーザー権限を示す定数の宣言
#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
#define DEFAULT_PITCH 0                    '文字ピッチデフォルト
#define PROOF_QUALITY 2                    'フォントの文字品質が、論理フォントの属性を正確に一致さ
                                           せることよりも重視
#define CLIP_DEFAULT_PRECIS 0
#define OUT_DEFAULT_PRECIS 0
```

```
#define FW_NORMAL 400
#define DEFAULT_CHARSET 1
#define TRANSPARENT 1                           '背景色を設定しない
#define LOGPIXELSX 88                           'スクリーン幅において1論理インチあたりのピクセル数
#define LOGPIXELSY 90                           'スクリーン高さにおいて1論理インチあたりのピクセル数
#define COLOR_WINDOW 5                          'ウィンドウの背景色
#define SRCCOPY &HCC0020                        'そのまま転送
#define MSG "北海道札幌市白石区菊水"

Var Shared hDC As Long
Var Shared mDC As Long
Var Shared mBitmap As Long

'=====================================
'=
'=====================================
Declare Function CreateMyFont(nSize As Integer, nDegrees As Long) As Long
Function CreateMyFont(nSize As Integer, nDegrees As Long) As Long
    CreateMyFont = Api_CreateFont(-Api_MulDiv(nSize, Api_GetDeviceCaps(Api_GetDC(0),
LOGPIXELSY), 72), 0, nDegrees * 10, 0, FW_NORMAL, 0, 0, 0, DEFAULT_CHARSET,
OUT_DEFAULT_PRECIS, CLIP_DEFAULT_PRECIS, PROOF_QUALITY, DEFAULT_PITCH, "MS 明朝")
End Function

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

    mDC = Api_CreateCompatibleDC(Api_GetDC(0))
    mBitmap = Api_CreateCompatibleBitmap(Api_GetDC(0), GetWidth, GetHeight)

    Ret = Api_SelectObject(mDC, mBitmap)
    Ret = Api_SetRect(rct, 0, 0, GetWidth, GetHeight)
    Ret = Api_FillRect(mDC, rct, Api_GetSysColorBrush(COLOR_WINDOW))

    Ret = Api_DeleteObject(Api_SelectObject(mDC, CreateMyFont(14, 15)))
    Ret = Api_TextOut(mDC, 5, GetHeight - 105, MSG, Len(MSG))

    Ret = Api_DeleteObject(Api_SelectObject(mDC, CreateMyFont(14, 0)))
    Ret = Api_TextOut(mDC, 5, GetHeight - 85, MSG, Len(MSG))
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    hDC = Api_GetDC(GethWnd)
    DrawText
End Sub

'=====================================
'= 印刷
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var DevName As String
    Var di As DOCINFO
    Var pd As PRINTER_DEFAULTS
    Var rct As RECT
    Var phDC As Long                            'プリンタのデバイスコンテキスト
    Var ex As Long
    Var Ret As Long

    'フォーム描画とプリントアウト時の倍率
    ex = 10
```

```
    '通常使うプリンタ(型式番号だけではなくプロパティでの名称)
    'プリンタ名の取得は、「プリンタの対応する用紙を取得」参照
    DevName = "pdfFactory Pro"
'   DevName = "EPSON PX-G920"
'   DevName = "KONICA MINOLTA magicolor 2400W"

    'プリンタ初期化
    pd.pDatatype = StrAdr(Chr$(0))
    pd.pDevMode = 0
    pd.DesiredAccess = PRINTER_ACCESS_ADMINISTER Or PRINTER_ACCESS_USE

    'プリンタオープン
    Ret = Api_OpenPrinter(DevName, phDC, pd)

    'プリンタデバイスコンテキストを取得
    phDC = Api_CreateDC("WINSPOOL", DevName, vbNullString, 0)
    If phDC = 0 Then GoTo *cleanup

    di.cbSize = Len(di)
    di.lpszOutput = StrAdr(Chr$(0))

    '印刷プロセス開始
    Ret = Api_StartDoc(phDC, di)
    Ret = Api_StartPage(phDC)

    '斜め15°の回転文字印刷
    Ret = Api_DeleteObject(Api_SelectObject(phDC, CreateMyFont(14 * ex, 15)))
    Ret = Api_TextOut(phDC, 5, (GetHeight - 105) * ex, MSG, Len(MSG))

    '水平の文字印刷
    Ret = Api_DeleteObject(Api_SelectObject(phDC, CreateMyFont(14 * ex, 0)))
    Ret = Api_TextOut(phDC, 5, (GetHeight - 85) * ex, MSG, Len(MSG))

    '印刷プロセス終了
    Ret = Api_EndPage(phDC)
    If Ret >= 0 Then Ret = Api_EndDoc(phDC)

*cleanup
    If phDC <> 0 Then Ret = Api_DeleteDC(phDC)
End Sub

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

    Ret = Api_BitBlt(hDC, 0, 0, GetWidth, GetHeight, mDC, 0, 0, SRCCOPY)
End Sub

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

画像転送

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
CreateBitmap 指定された幅・高さ・色形式を持つビットマップを作成
SetBkColor デバイスコンテキストの背景色を設定
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

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
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
DeleteDC 指定されたデバイスコンテキストを削除
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
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

Picture2の画像をPicture1に転送しています。


```
'========================================
'= 画像転送
'=    (BitBlt.bas)
'========================================
#include "Windows.bi"

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

' 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&)

' 指定された幅・高さ・色形式を持つビットマップを作成
Declare Function Api_CreateBitmap& Lib "gdi32" Alias "CreateBitmap" (ByVal nWidth&, ByVal nHeight&, ByVal nPlanes&, ByVal nBitCount&, lpBits As Any)

' 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&)

' 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 SRCCOPY &HCC0020                    '単純な上書き
```

```
#define NOTSRCCOPY &H330008                    '反転コピー
#define SRCAND &H8800C6                        '転送先の画像とAND演算して転送
#define SRCINVERT &H66004                      '転送先の画像とXOR演算して転送
#define vbWhite &HFFFFFF                       '白のカラーコード

Var Shared PICTURE(2) As Object
Var Shared Bitmap As Object

For i = 0 To 2
    Picture(i).Attach GetDlgItem("Picture" & Trim$(Str$(i + 1)))
Next i
BitmapObject Bitmap

Var Shared rc As RECT
Var Shared pMove As Integer
Var Shared hDC1 As Long
Var Shared hDC2 As Long

'=====================================
'=
'=====================================
Declare Sub TranspPic(OutDstDC As Long, DstDC As Long, SrcDC As Long, SrcRect As RECT,
ByVal DstX As Long, ByVal DstY As Long, TransColor As Long)
Sub TranspPic(OutDstDC As Long, DstDC As Long, SrcDC As Long, SrcRect As RECT, ByVal DstX
As Long, ByVal DstY As Long, TransColor As Long)
    Var W As Long
    Var H As Long
    Var MonoMaskDC As Long
    Var hMonoMask As Long
    Var MonoInvDC As Long
    Var hMonoInv As Long
    Var ResultDstDC As Long
    Var hResultDst As Long
    Var ResultSrcDC As Long
    Var hResultSrc As Long
    Var hPrevMask As Long
    Var hPrevInv As Long
    Var hPrevSrc As Long
    Var hPrevDst As Long
    Var OldBC As Long
    Var Ret As Long
    W = SrcRect.Right - SrcRect.Left
    H = SrcRect.Bottom - SrcRect.Top

    'モノクロと反転マスク
    MonoMaskDC = Api_CreateCompatibleDC(DstDC)
    MonoInvDC = Api_CreateCompatibleDC(DstDC)
    hMonoMask = Api_CreateBitmap(W, H, 1, 1, ByVal 0)
    hMonoInv = Api_CreateBitmap(W, H, 1, 1, ByVal 0)
    hPrevMask = Api_SelectObject(MonoMaskDC, hMonoMask)
    hPrevInv = Api_SelectObject(MonoInvDC, hMonoInv)

    'バッファ作成
    ResultDstDC = Api_CreateCompatibleDC(DstDC)
    ResultSrcDC = Api_CreateCompatibleDC(DstDC)
    hResultDst = Api_CreateCompatibleBitmap(DstDC, W, H)
    hResultSrc = Api_CreateCompatibleBitmap(DstDC, W, H)
    hPrevDst = Api_SelectObject(ResultDstDC, hResultDst)
    hPrevSrc = Api_SelectObject(ResultSrcDC, hResultSrc)

    'モノクロマスク
    OldBC = Api_SetBkColor(SrcDC, TransColor)
    Ret = Api_BitBlt(MonoMaskDC, 0, 0, W, H, SrcDC, SrcRect.Left, SrcRect.Top, SRCCOPY)
    TransColor = Api_SetBkColor(SrcDC, OldBC)

    '反転作成
    Ret = Api_BitBlt(MonoInvDC, 0, 0, W, H, MonoMaskDC, 0, 0, NOTSRCCOPY)

    '最終画像の背景
    Ret = Api_BitBlt(ResultDstDC, 0, 0, W, H, DstDC, DstX, DstY, SRCCOPY)
```

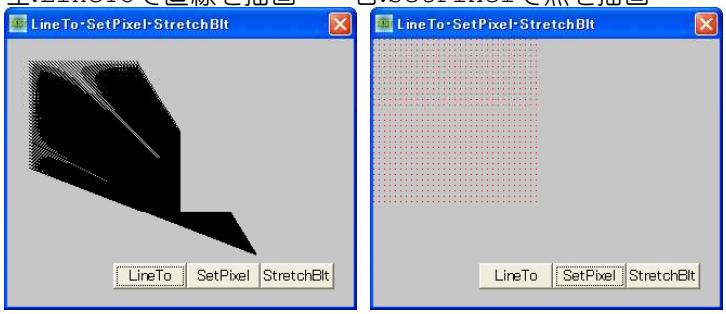
```
    'マスクをAND合成
    Ret = Api_BitBlt(ResultDstDC, 0, 0, W, H, MonoMaskDC, 0, 0, SRCAND)

    '
    Ret = Api_BitBlt(ResultSrcDC, 0, 0, W, H, SrcDC, SrcRect.Left, SrcRect.Top, SRCCOPY)

    '反転したモノクロマスクのAND合成
    Ret = Api_BitBlt(ResultSrcDC, 0, 0, W, H, MonoInvDC, 0, 0, SRCAND)

    'XOR合成
    Ret = Api_BitBlt(ResultDstDC, 0, 0, W, H, ResultSrcDC, 0, 0, SRCINVERT)

    '最終コピー結果
    Ret = Api_BitBlt(OutDstDC, DstX, DstY, W, H, ResultDstDC, 0, 0, SRCCOPY)

    'オブジェクトの生成、DC解放
    hMonoMask = Api_SelectObject(MonoMaskDC, hPrevMask)
    Ret = Api_DeleteObject(hMonoMask)
    Ret = Api_DeleteDC(MonoMaskDC)

    hMonoInv = Api_SelectObject(MonoInvDC, hPrevInv)
    Ret = Api_DeleteObject(hMonoInv)
    Ret = Api_DeleteDC(MonoInvDC)

    hResultDst = Api_SelectObject(ResultDstDC, hPrevDst)
    Ret = Api_DeleteObject(hResultDst)
    Ret = Api_DeleteDC(ResultDstDC)

    hResultSrc = Api_SelectObject(ResultSrcDC, hPrevSrc)
    Ret = Api_DeleteObject(hResultSrc)
    Ret = Api_DeleteDC(ResultSrcDC)
End Sub

'=====================================
'=
'=====================================
Declare Sub MovePicTo (ByVal X As Long, ByVal Y As Long)
Sub MovePicTo(ByVal X As Long, ByVal Y As Long)
    X = X - rc.Right / 2
    Y = Y - rc.Bottom / 2
    Bitmap.LoadFile "Flower256.bmp"
    Picture(0).DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject
    TranspPic hDC1, hDC2, hDC2, rc, X, Y, vbWhite
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Bitmap.LoadFile "flower256.bmp"
    Picture(0).DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject

    Bitmap.LoadFile "bike6.bmp"            '"flower_Gray.bmp"
    Picture(1).DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject

    hDC1 = Api_GetDC(Picture(0).GethWnd)
    hDC2 = Api_GetDC(Picture(1).GethWnd)

    rc.Left = 0
    rc.Top = 0
    rc.Right = Picture(1).GetWidth
    rc.Bottom = Picture(1).GetHeight
End Sub
```

```
'=====================================
'=
'=====================================
Declare Sub Picture1_Click edecl ()
Sub Picture1_Click()
    If pMove = False Then
        pMove = True
    Else
        pMove = False
    End If
End Sub

'=====================================
'=
'=====================================
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)
    If pMove Then
        MovePicTo X, Y
    End If
End Sub

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

    Ret = Api_ReleaseDC(Picture(0).GethWnd, hDC1)
    Ret = Api_ReleaseDC(Picture(1).GethWnd, hDC2)
End Sub

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

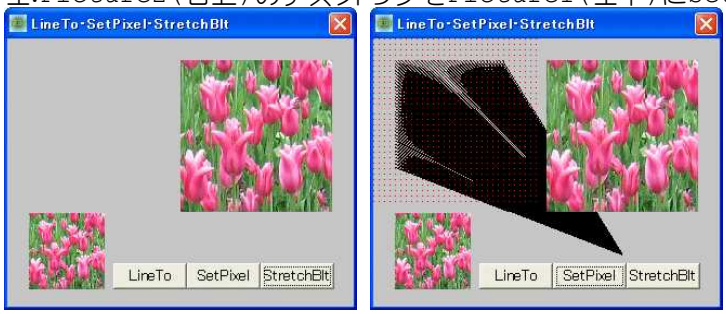
画像転送とピクセル描画・直線描画

既出のテストと重複しますが、拡大縮小転送・ピクセル描画・直線描画を実行しています。
SetPixel 指定した座標に点を配置する
PaintDesktop 指定の領域をデスクトップと同じパターン・壁紙で塗りつぶす
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
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
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

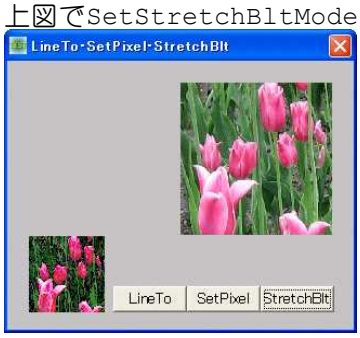
左:LineToで直線を描画　　右:SetPixelで点を描画



左:Picture2(右上)のデスクトップをPicture1(左下)にSetStretchBltModeで転送 右:全てのボタンをクリック



上図でSetStretchBltModeを使用しない場合の転送先画像




```
'=====================================
'= 画像転送とピクセル描画・直線描画
'=    (SetPixel.bas)
'=====================================
#include "Windows.bi"

Type POINTAPI
    X   As Long
    Y   As Long
End Type

' 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&)

' 指定の領域をデスクトップと同じパターン・壁紙で塗りつぶす
Declare Function Api_PaintDesktop& Lib "user32" Alias "PaintDesktop" (ByVal hDC&)

' 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&)

' 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&)

' 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&, lpPoint As POINTAPI)

' 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
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
#define COLORONCOLOR 3

Var Shared Picture1 As Object
Var Shared Picture2 As Object
```

```
Picture1.Attach GetDlgItem("Picture1")
Picture2.Attach GetDlgItem("Picture2")

Var Shared fmhDC As Long
Var Shared p1hDC As Long
Var Shared p2hDC As Long

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Picture2.SetWindowSize 140, 140        'PictureBoxサイズ指定
    Picture1.SetWindowSize 70, 70

    fmhDC = Api_GetDC(GethWnd)
    p1hDC = Api_GetDC(Picture1.GethWnd)
    p2hDC = Api_GetDC(Picture2.GethWnd)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var i As byte
    Var j As byte
    Var nPoint As POINTAPI
    Var Ret As Long

    For i = 20 To 120 step 3
        For j = 20 To 120 step 3
            nPoint.X = i                           'スタートポイント設定
            nPoint.Y = j

            Ret = Api_MoveToEx(fmhDC, i, j, nPoint) 'アクティブポイント
            Ret = Api_LineTo(fmhDC, 230, 200)       'アクティブポイントから線を引く
        Next j
    Next i
End Sub

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

    For i = 0 To 150 step 5
        For j = 0 To 150 Step 5
            Ret = Api_SetPixel(fmhDC, i, j, RGB(255, 0, 0)) 'ピクセル描画
        Next j
    Next i
End Sub

'=====================================
'=
'=====================================
Declare Sub Button3_on edecl ()
Sub Button3_on()
    Var Ret As Long
    Ret = Api_PaintDesktop(p2hDC)          'デスクトップをコピー
    Ret = Api_SetStretchBltMode(p1hDC, COLORONCOLOR)
    Ret = Api_StretchBlt(p1hDC, 0, 0, 70, 70, p2hDC, 0, 0, 140, 140, SRCCOPY) '拡大伸縮転送
End Sub
```

```
'=====================================
'=
'=====================================
Declare Sub MainFOrm_QueryClose edecl (Cancel As Integer, Mode As Integer)
Sub MainForm_QueryClose(Cancel As Integer, Mode As Integer)
    Var Ret As Long

    If cancel = 0 Then
        Ret = Api_ReleaseDC(GethWnd, fmDC)
        Ret = Api_ReleaseDC(Picture1.GethWnd, ph1DC)
        Ret = Api_ReleaseDC(Picture2.GethWnd, ph2DC)
    End If
End Sub

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

画像透過転送

画像を透過転送します。
BitBlt ビットブロック転送
CreateBitmap 指定された幅・高さ・色形式を持つビットマップを作成
SetBkColor デバイスコンテキストの背景色を設定
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
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
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
DeleteObject 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


```
'=====================================
'= 画像透過転送
'=    (TransparentBlt2.bas)
'=====================================
#include "Windows.bi"

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

```
End Type

' 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&)

' 指定された幅・高さ・色形式を持つビットマップを作成
Declare Function Api_CreateBitmap& Lib "gdi32" Alias "CreateBitmap" (ByVal nWidth&, ByVal nHeight&, ByVal nPlanes&, ByVal nBitCount&, lpBits As Any)

' 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&)

' 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 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
#define NOTSRCCOPY &H330008             '転送元の色データが全反転
#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 vbWhite &HFFFFFF                '白のカラーコード
#define vbBlack &H000000                '黒のカラーコード

Var shared Picture1 As Object
Var shared Radio(3) As Object
Var Shared Button1 As Object
Var shared Bitmap As Object

Picture1.Attach GetDlgItem("Picture1")
For i = 0 To 3
    Radio(i).Attach GetDlgItem("Radio" & Trim$(Str$(i + 1))) : Radio(i).SetFontSize 14
Next i
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
BitmapObject Bitmap

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

```
Declare Sub TransparentBlt(OutDstDC As Long, DstDC As Long, SrcDC As Long, SrcRect As
RECT, DstX As Integer, DstY As Integer, TransColor As Long)
Sub TransparentBlt(OutDstDC As Long, DstDC As Long, SrcDC As Long, SrcRect As RECT, DstX As
Integer, DstY As Integer, TransColor As Long)
    Var W As Integer
    Var H As Integer
    Var MonoMaskDC As Long
    Var hMonoMask As Long
    Var MonoInvDC As Long
    Var hMonoInv As Long
    Var ResultDstDC As Long
    Var hResultDst As Long
    Var ResultSrcDC As Long
    Var hResultSrc As Long
    Var hPrevMask As Long
    Var hPrevInv As Long
    Var hPrevSrc As Long
    Var hPrevDst As Long
    Var OldBC As Long
    Var Ret As Long

    W = SrcRect.Right - SrcRect.Left + 1
    H = SrcRect.Bottom - SrcRect.Top + 1

    'モノクロマスクおよびインバートマスクの作成
    MonoMaskDC = Api_CreateCompatibleDC(DstDC)
    MonoInvDC = Api_CreateCompatibleDC(DstDC)
    hMonoMask = Api_CreateBitmap(W, H, 1, 1, ByVal 0)
    hMonoInv = Api_CreateBitmap(W, H, 1, 1, ByVal 0)
    hPrevMask = Api_SelectObject(MonoMaskDC, hMonoMask)
    hPrevInv = Api_SelectObject(MonoInvDC, hMonoInv)

    'デバイスコンテキストとビットマップの作成
    ResultDstDC = Api_CreateCompatibleDC(DstDC)
    ResultSrcDC = Api_CreateCompatibleDC(DstDC)
    hResultDst = Api_CreateCompatibleBitmap(DstDC, W, H)
    hResultSrc = Api_CreateCompatibleBitmap(DstDC, W, H)
    hPrevDst = Api_SelectObject(ResultDstDC, hResultDst)
    hPrevSrc = Api_SelectObject(ResultSrcDC, hResultSrc)

    'モノクロマスクのコピー
    OldBC = Api_SetBkColor(SrcDC, TransColor)
    nRet = Api_BitBlt(MonoMaskDC, 0, 0, W, H, SrcDC, SrcRect.Left, SrcRect.Top, SRCCOPY)
    TransColor = Api_SetBkColor(SrcDC, OldBC)

    'マスクのインバース作成
    Ret = Api_BitBlt(MonoInvDC, 0, 0, W, H, MonoMaskDC, 0, 0, NOTSRCCOPY)

    'バックグラウンドの作成
    Ret = Api_BitBlt(ResultDstDC, 0, 0, W, H, DstDC, DstX, DstY, SRCCOPY)

    'モノクロマスクとのAND演算
    Ret = Api_BitBlt(ResultDstDC, 0, 0, W, H, MonoMaskDC, 0, 0, SRCAND)

    'オーバーラップの作成
    Ret = Api_BitBlt(ResultSrcDC, 0, 0, W, H, SrcDC, SrcRect.Left, SrcRect.Top, SRCCOPY)

    'インバースモノクロマスクとのAND演算
    Ret = Api_BitBlt(ResultSrcDC, 0, 0, W, H, MonoInvDC, 0, 0, SRCAND)

    '二つのXOR演算
    Ret = Api_BitBlt(ResultDstDC, 0, 0, W, H, ResultSrcDC, 0, 0, SRCINVERT)

    '出力
    Ret = Api_BitBlt(OutDstDC, DstX, DstY, W, H, ResultDstDC, 0, 0, SRCCOPY)

    'クリーンアップ
    hMonoMask = Api_SelectObject(MonoMaskDC, hPrevMask)
    Ret = Api_DeleteObject(hMonoMask)
```

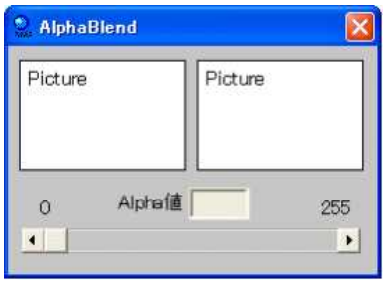
```
    hMonoInv = Api_SelectObject(MonoInvDC, hPrevInv)
    Ret = Api_DeleteObject(hMonoInv)

    hResultDst = Api_SelectObject(ResultDstDC, hPrevDst)
    Ret = Api_DeleteObject(hResultDst)

    hResultSrc = Api_SelectObject(ResultSrcDC, hPrevSrc)
    Ret = Api_DeleteObject(hResultSrc)

    Ret = Api_DeleteDC(MonoMaskDC)
    Ret = Api_DeleteDC(MonoInvDC)
    Ret = Api_DeleteDC(ResultDstDC)
    Ret = Api_DeleteDC(ResultSrcDC)
End Sub

'=====================================
'=
'=====================================
Declare Sub Radio1_on edecl ()
Sub Radio1_on()
    Bitmap.LoadFile "Bike1.bmp"
    Picture1.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject
    Cls
End Sub

'=====================================
'=
'=====================================
Declare Sub Radio2_on edecl ()
Sub Radio2_on()
    Bitmap.LoadFile "Bike5.bmp"
    Picture1.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject
    Cls
End Sub

'=====================================
'=
'=====================================
Declare Sub Radio3_on edecl ()
Sub Radio3_on()
    Cls
End Sub

'=====================================
'=
'=====================================
Declare Sub Radio4_on edecl ()
Sub Radio4_on()
    Cls
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start
    Radio1_on
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var rc As RECT
    Var phDC As Long
    Var fhDC As Long
    Var Mask As Long
```

```
    Var Ret As Long

    phDC = Api_GetDC(Picture1.GethWnd)
    fhDC = Api_GetDC(GethWnd)

    rc.Left = 0
    rc.Top = 0
    rc.Right = Picture1.GetWidth
    rc.Bottom = Picture1.GetHeight

    cls
    If Radio(2).GetCheck = 1 Then
        Mask = vbWhite
    Else If Radio(3).GetCheck = 1 Then
        Mask = vbBlack
    End If

    TransparentBlt fhDC, fhDC, phDC, rc, 30, 60, Mask

    Ret = Api_ReleaseDC(phDC, Picture1.GethWnd)
    Ret = Api_ReleaseDC(fhDC, GethWnd)
End Sub

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

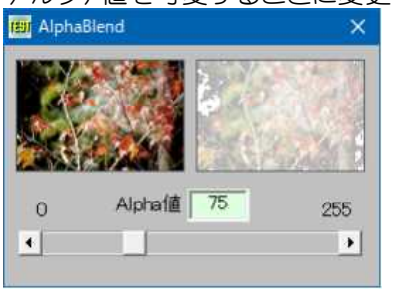
画像透過転送(Ⅰ)

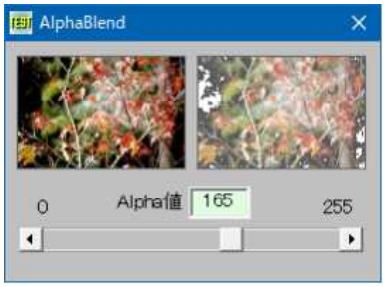
AlphaBlendを使って画像を転送します。
あらかじめ120×80ピクセルのBMPファイルをAlphaTest.bmpとして用意してください。
AlphaBlend 画像転送
GetDC デバイスコンテキスト取得
ReleaseDC デバイスコンテキスト解放

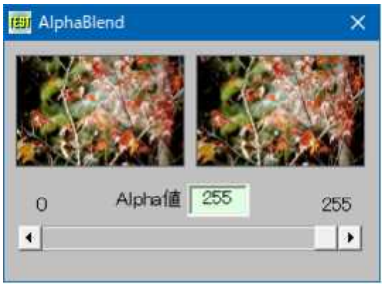
フォーム設計



アルファ値を可変することに変更しました。







アルファ値 【Alpha value】
コンピュータが扱うデジタル画像データにおいて、各点に設定された透過度情報のこと。完全な透明（無色）から、完全な不透明（背景の色をまったく通さない）まで設定することができる。
コンピュータが画像を扱う場合、色情報として、各点についてR（赤）・G（緑）・B（青）の三原色の情報をもち、その組み合わせで色を表現する（CMYKモードの場合は4色）。
点の透明度を表現する場合にはこれにアルファ値を加え、4つの情報の組み合わせで一つの点を表現する。
アルファ値は、データ形式やソフトウェアによって扱える場合と扱えない場合がある。扱える場合、各点の色を表すデータに追加する形で表現される。この追加されたデータ領域をアルファチャンネルという。（この項**e-Word**を参照しました）

```
'=====================================
'= AlphaBlEndで透過転送
'=    (AlphaBlend.bas)
'=====================================
#include "Windows.bi"

' 透過ピクセルと半透過ピクセルを持つビットマップを表示
Declare Function Api_AlphaBlend& Lib "msimg32" Alias "AlphaBlend" (ByVal hdcDest&, ByVal
nXDest&, ByVal nYDest&, ByVal nWidthDest&, ByVal nHeightDest&, ByVal hdcSrc&, ByVal
nXSrc&, ByVal nYSrc&, ByVal nWidthSrc&, ByVal nHeightSrc&, ByVal nBlendFunc&)

' 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 HScroll1 As Object
Var Shared Picture1 As Object
Var Shared Picture2 As Object
Var Shared Text(3) As Object
Var Shared Bitmap As Object
BitmapObject Bitmap

HScroll1.Attach GetDlgItem("HScroll1")
Picture1.Attach GetDlgItem("Picture1")
Picture2.Attach GetDlgItem("Picture2")
For i = 0 To 3
    Text(i).Attach GetDlgItem("Text" & Trim$(Str$(i + 1)))
    Text(i).SetFontSize 14
Next i

Var Shared hDC1 As Long
Var Shared hDC2 As Long

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    HScroll1.SetScrollRange 0,255
    HScroll1.SetScrollStep 5, 1
    HScroll1.SetScrollPos 0

    Bitmap.LOADFILE "AlphaTest.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

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

```
Declare Sub HScroll1_Change edecl ()
Sub HScroll1_Change()
    Var Alpha As Long
    Var Ret As Long

    Picture2.Cls
    Alpha = HScroll1.GetScrollPos
    Ret = Api_AlphaBlend(hDC2, 0, 0, Picture2.GetWidth, Picture2.GetHeight , hDC1, 0, 0,
Picture1.GetWidth, Picture1.GetHeight , Alpha * &H10000)
    Text(3).SetWindowText Trim$(Str$(Alpha))
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
```

画像透過転送(II)フェードインもどき？

AlphaBlendを使って画像を転送します。
あらかじめ120×80ピクセルのBMPファイルをALPHATEST.BMP・ALPHATEST2.BMPとして用意してください。
AlphaBlend 画像転送
GetDC コンテキスト取得
ReleaseDC コンテキスト解放

Picture2がPicture1の画像に変化します。
見た目には、Alpha値がおよそ30位でほぼ転送完了に見えるため、30以下にWaitを挿入しています。


```
'=====================================
'= 画像透過転送(II)
'=    フェードイン擬き
'=    (AlphaBlend2.bas)
'=====================================
#include "Windows.bi"

' 透過ピクセルと半透過ピクセルを持つビットマップを表示
Declare Function Api_AlphaBlend& Lib "msimg32" Alias "AlphaBlend" (ByVal hdcDest&, ByVal
nXDest&, ByVal nYDest&, ByVal nWidthDest&, ByVal nHeightDest&, ByVal hdcSrc&, ByVal
nXSrc&, ByVal nYSrc&, ByVal nWidthSrc&, ByVal nHeightSrc&, ByVal nBlendFunc&)
```

```
' 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
Var shared Picture2 As Object
Var shared Bitmap As Object
Var shared Text1 As Object
BitmapObject Bitmap

Picture1.Attach GetDlgItem("Picture1")
Picture2.Attach GetDlgItem("Picture2")
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var shared hDC1 As Long
Var shared hDC2 As Long

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Bitmap.LoadFile "ALPHATEST.BMP"
    Picture1.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject

    Bitmap.LoadFile "ALPHATEST2.BMP"
    Picture2.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject

    hDC1 = Api_GetDC(Picture1.GethWnd)
    hDC2 = Api_GetDC(Picture2.GethWnd)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var i As Long
    Var Ret As Long
    Button1.EnableWindow 0
    For i = 0 To 255
        Ret = Api_AlphaBlend(hDC2, 0, 0, Picture2.GetWidth, Picture2.GetHeight, hDC1, 0,
0, Picture1.GetWidth, Picture1.GetHeight, i * &H10000)
        If i < 30 Then Wait 10
        Text1.SetWindowText "i =" & Str$(i)
    Next

    Text1.SetWindowText ""
    Picture2.Refresh
    Button1.EnableWindow -1
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
```

画像透過転送(III)

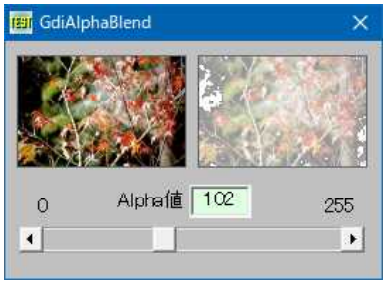
AlphaBlend 透過ピクセルと半透過ピクセルを持つビットマップを表示
MoveMemory メモリの指定領域をコピー
GetDC 指定されたウィンドウのデバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放


```
'=====================================
'= 画像透過転送(III)
'=    (AlphaBlend3.bas)
'=====================================
#include "Windows.bi"

#define AC_SRC_ALPHA &H1                    '
#define AC_SRC_OVER &H0                     '

Type BLENDFUNCTION
    BlendOp           As Byte               'ブレンド操作(AC_SRC_OVER)
    BlendFlags        As Byte               '常に0
    SourceConstantAlpha As Byte             'コピー元のビットマップ全体に適用するアルファ値
    AlphaFormat       As Byte               'コピー元のビットマップがアルファ値を持つとき
                                            (AC_SRC_ALPHA)
End Type

' 透過ピクセルと半透過ピクセルを持つビットマップを表示
Declare Function Api_AlphaBlend& Lib "msimg32" Alias "AlphaBlend" (ByVal hdcDest&, ByVal
nXDest&, ByVal nYDest&, ByVal nWidthDest&, ByVal nHeightDest&, ByVal hdcSrc&, ByVal
nXSrc&, ByVal nYSrc&, ByVal nWidthSrc&, ByVal nHeightSrc&, ByVal nBlendFunc&)

' 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&)

' 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 Picture2 As Object
Var Shared Button1 As Object
Var Shared Bitmap As Object

Picture1.Attach GetDlgItem("Picture1")
Picture2.Attach GetDlgItem("Picture2")
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

    Bitmap.LoadFile "Bike1.bmp"
    Picture2.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject
End Sub

'====================================
'=
'====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var phDC1 As Long
    Var phDC2 As Long
    Var bf As BLENDFUNCTION
    Var lBF As Long
    Var Alpha As Byte
    Var Ret As Long

    phDC1 = Api_GetDC(Picture1.GethWnd)
    phDC2 = Api_GetDC(Picture2.GethWnd)

    Alpha = 180

    bf.BlendOp = AC_SRC_OVER
    bf.BlendFlags = 0
    bf.SourceConstantAlpha = Alpha
    bf.AlphaFormat = 0

    MoveMemory lBF, BF, 4

    Ret = Api_AlphaBlend(phDC1, 10, 10, 50, 50, phDC2, 0, 0, Picture2.GetWidth,
Picture2.GetHeight, lBF)

    Ret = Api_ReleaseDC(Picture1.GethWnd, phDC1)
    Ret = Api_ReleaseDC(Picture2.GethWnd, phDC1)
End Sub

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

画像透過転送(GdiAlphaBlend)

GdiAlphaBlendを使って画像を転送します。
あらかじめ120×80ピクセルのBMPファイルをALPHATEST.BMPとして用意してください。
GdiAlphaBlend 画像転送
RtlMoveMemory メモリブロックのコピー
GetDC コンテキスト取得
ReleaseDC コンテキスト解放




```
'=====================================
'= 透過転送(GdiAlphaBlend)
'=    (GdiAlphaBlend.bas)
'=====================================
#include "Windows.bi"

Type BLENDFUNCTION
    BlendOp                 As Byte
    BlendFlags              As Byte
    SourceConstantAlpha     As Byte
    AlphaFormat             As Byte
End Type
' 画像を透過転送
Declare Function Api_GdiAlphaBlend& Lib "gdi32" Alias "GdiAlphaBlend" (ByVal hDC&, ByVal
lInt&, ByVal lInt&, ByVal lInt&, ByVal lInt&, ByVal hDC&, ByVal lInt&, ByVal lInt&, ByVal
lInt&, ByVal lInt&, ByVal BLENDFUNCT&)

' 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&)

' 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 HScroll1 As Object
Var shared Picture1 As Object
Var shared Picture2 As Object
Var shared Text4 As Object
Var shared Bitmap As Object
BitmapObject Bitmap

HScroll1.Attach GetDlgItem("HScroll1")
Picture1.Attach GetDlgItem("Picture1")
Picture2.Attach GetDlgItem("Picture2")
Text4.Attach GetDlgItem("Text4")

Var shared hDC1 As Long
Var shared hDC2 As Long

'=====================================
'=
'=====================================
Declare Sub Mainform_Start edecl ()
Sub Mainform_Start()
    HScroll1.SetScrollRange 0,255
    HScroll1.SetScrollStep 5, 1
    HScroll1.SetScrollPos 0

    Bitmap.LoadFile "AlphaTest.Bmp"
    Picture1.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject

    hDC1 = Api_GetDC(Picture1.GethWnd)
    hDC2 = Api_GetDC(Picture2.GethWnd)
End Sub
```

```
'=====================================
'=
'=====================================
Declare Sub HScroll1_Change edecl ()
Sub HScroll1_Change()
    Var BF As BLENDFUNCTION
    Var Alpha As Long
    Var lBF As Long
    Var Res As Long

    Picture2.Cls
    Alpha = HScroll1.GetScrollPos

    BF.BlendOp = AC_SRC_OVER
    BF.BlendFlags = 0
    BF.SourceConstantAlpha = Alpha
    BF.AlphaFormat = 0

    '構造体のコピー
    CopyMemory lBF, BF, 4

    '画像透過転送
    Res = Api_GdiAlphaBlend(hDC2, 0, 0, Picture2.GetWidth, Picture2.GetHeight, hDC1, 0,
0, Picture1.GetWidth, Picture1.GetHeight, lBF)

    Text4.SetWindowText Trim$(Str$(Alpha))
End Sub

'=====================================
'=
'=====================================
Declare Sub Mainform_QueryClose edecl (Cancel%, ByVal Mode%)
Sub Mainform_QueryClose(Cancel%, ByVal Mode%)
    Var Res As Long

    If Cancel% = 0 Then
        Res = Api_ReleaseDC(Picture1.GethWnd, hDC1)
        Res = Api_ReleaseDC(Picture2.GethWnd, hDC2)
        End
    End If
End Sub

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

画像透過転送(GdiTransparentBlt)

TransParentBltによる透過転送と同じです。Windows2000以降(Windows98/Meはサポート外)
GdiTransparentBlt 透明色を指定してビットマップをコピー

左:白で透過転送　右:黒で透過転送

```
'====================================
'= 透過転送テスト(GdiTransparentBlt)
'=    (GdiTransparentBlt.bas)
'====================================
#include "Windows.bi"

' 透明色を指定してビットマップをコピー
Declare Function Api_GdiTransparentBlt& Lib "gdi32" Alias "GdiTransparentBlt" (ByVal
hDC&, ByVal x&, ByVal y&, ByVal nWidth&, ByVal nHeight&, ByVal hSrcDC&, ByVal xSrc&, ByVal
ySrc&, ByVal nSrcWidth&, ByVal nSrcHeight&, ByVal crTransparent&)

' 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 Bitmap As Object
Var Shared Mainform As Object
Var Shared Picture1 As Object
Var Shared Picture2 As Object
Var Shared Picture3 As Object

Mainform.Attach GethWnd
Picture1.Attach GetDlgItem("Picture1")
Picture2.Attach GetDlgItem("Picture2")
Picture3.Attach GetDlgItem("Picture3")
BitmapObject Bitmap

Var Shared RGBColor As Long                    '透明色
Var Shared hDC0 As Long
Var Shared hDC1 As Long
Var Shared hDC2 As Long
Var Shared hDC3 As Long
Var Shared xPos As Integer

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

    Bitmap.LoadFile "BackColor2.bmp"
    Picture1.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject

    Bitmap.LoadFile "bike1.bmp"
    Picture2.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject

    Bitmap.LoadFile "bike5.bmp"
    Picture3.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject

    hDC1 = Api_GetDC(Picture1.GethWnd)
End Sub

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

    Ret = Api_GdiTransparentBlt(hDC1, xPos, 10, 56, 52, hDC0, 0, 0, 56, 52, RGBColor)
End Sub

'====================================
'=  透明色を白
'====================================
```

```
Declare Sub Button1_on edecl ()
Sub Button1_on()
    RGBColor = RGB(255, 255, 255)
    hDC2 = Api_GetDC(Picture2.GethWnd)
    hDC0 = hDC2
    xPos = 150
    Tensou_dsp
End Sub

'=======================================
'=  透明色を黒
'=======================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    RGBColor = RGB(0, 0, 0)
    hDC3 = Api_GetDC(Picture3.GethWnd)
    hDC0 = hDC3
    xPos = 250
    Tensou_dsp
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)
        Ret = Api_ReleaseDC(Picture3.GethWnd, hDC3)
        End
    End If
End Sub

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

画像透過転送(TransparentBlt)

透過転送をテストします。
TransparentBlt 透明色を指定してビットマップをコピー
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

白を透明色に指定した場合

黒を透明色に指定した場合

```
'=====================================
'= TransparentBltによる透過転送テスト
'=    (TransparentBlt.bas)
'=====================================
#include "Windows.bi"

' 透明色を指定してビットマップをコピー
Declare Function Api_TransparentBlt& Lib "msimg32" Alias "TransparentBlt" (ByVal hDC&,
ByVal x&, ByVal y&, ByVal nWidth&, ByVal nHeight&, ByVal hSrcDC&, ByVal xSrc&, ByVal
ySrc&, ByVal nSrcWidth&, ByVal nSrcHeight&, ByVal crTransparent&)

' 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 Bitmap As Object
Var shared Picture1 As Object
Var shared Picture2 As Object
Var shared Picture3 As Object
Var Shared Button1 As Object
Var Shared Button2 As Object
Picture1.Attach GetDlgItem("Picture1")
Picture2.Attach GetDlgItem("Picture2")
Picture3.Attach GetDlgItem("Picture3")
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14
BitmapObject Bitmap

Var shared RGBColor As Long                    '透明色
Var shared hDC0 As Long
Var shared hDC1 As Long
Var shared hDC2 As Long
Var shared hDC3 As Long
Var shared xPos As Integer

'=====================================
'=
'=====================================
Declare Sub Mainform_Start edecl ()
Sub Mainform_Start()
    Bitmap.LoadFile "BackColor2.bmp"
    Picture1.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject

    Bitmap.LoadFile "bike1.bmp"
    Picture2.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject

    Bitmap.LoadFile "bike5.bmp"
    Picture3.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject

    hDC1 = Api_GetDC(Picture1.GethWnd)
End Sub

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

    Ret = Api_TransparentBlt(hDC1, xPos, 10, 56, 52, hDC0, 0, 0, 56, 52, RGBColor)
End Sub

'=====================================
'=  透明色を白
'=====================================
```

```
Declare Sub Button1_on edecl ()
Sub Button1_on()
    RGBColor = RGB(255, 255, 255)
    hDC2 = Api_GetDC(Picture2.GethWnd)
    hDC0 = hDC2
    xPos = 150
    Tensou_dsp
End Sub

'=====================================
'=  透明色を黒
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    RGBColor = RGB(0, 0, 0)
    hDC3 = Api_GetDC(Picture3.GethWnd)
    hDC0 = hDC3
    xPos = 250
    Tensou_dsp
End Sub

'=====================================
'=
'=====================================
Declare Sub Mainform_QueryClose edecl (Cancel%, ByVal Mode%)
Sub Mainform_QueryClose(Cancel%, ByVal Mode%)
    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
```

画像のネガ・ポジ変換（Ⅰ）

写真、絵画のネガ・ポジの変換を実行します。
BitBlt ビットブロックの転送
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
DSTINVERT(&H550009) 反転してコピー

ボタンをクリックするたびにネガ・ポジを繰り返します。

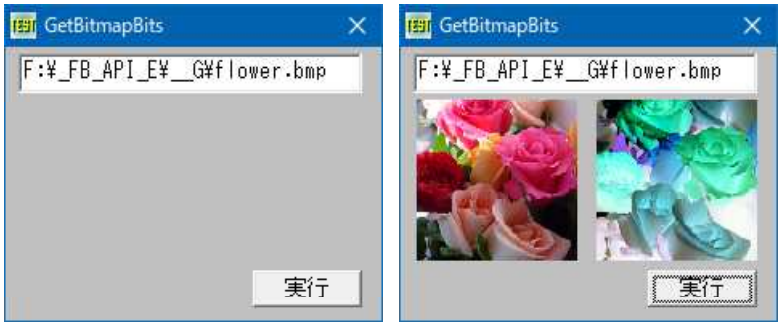

```
'=====================================
'= ネガ・ポジ変換
'=    (Nega.bas)
'=====================================
#include "Windows.bi"
```

```
' 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 DSTINVERT &H550009                    'コピー先を反転してコピー

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.StretchBitmap Bitmap, 0, 0, Picture1.GetWidth, Picture1.GetHeight
    Bitmap.DeleteBitmap
End Sub

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

    hDC = Api_GetDC(Picture1.GethWnd)

    'Positive <-> Negative
    Ret = Api_BitBlt(hDC, 0, 0, Picture1.GetWidth, Picture1.GetHeight, hDC, 0, 0, DSTINVERT)

    Ret = Api_ReleaseDC(Picture1.GethWnd, hDC)
End Sub

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

画像のネガ・ポジ変換（II）

LoadImage 画像ファイルの読み込み
GetObject オブジェクト取得
GetBitmapBits 指定されたビットマップのビットを取得し、バッファにコピー
SetBitmapBits ビットマップに指定された値をセット
BitBlt ビットブロック転送
OpenClipboard クリップボードをオープン
EmptyClipboard クリップボードを空にする
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

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
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
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
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、取り込んだBitmapを反転後、クリップボードに転送しPicture2に貼り付けています。

```
'=======================================
'= 画像のポジ・ネガ変換(II)
'=    (GetBitmapBits2.bas)
'=======================================
#include "Windows.bi"

Type BITMAP
    bmType          As Long             'ビットマップ タイプを指定(0)
    bmWidth         As Long             'ビットマップの幅を指定
    bmHeight        As Long             'ビットマップの高さを指定
    bmWidthBytes    As Long             '1 走査線あたりのバイト数を指定
    bmPlanes        As Integer          'カラープレーンを指定(通常1)
    bmBitsPixel     As Integer          '1 ピクセルを定義するのに必要なビット数を指定
    bmBits          As Long             'ビット データが格納されているの配列へのポインタを指定
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
#define CF_BITMAP 2                     'ビットマップのデータ(HBITMAP)
#define CrLf Chr$(13, 10)

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

' 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&)

' 指定されたビットマップのビットを取得し、バッファにコピー
Declare Function Api_GetBitmapBits& Lib "gdi32" Alias "GetBitmapBits" (ByVal hBitmap&,
ByVal dwCount&, lpBits As Any)

' ビットマップに指定された値をセット
Declare Function Api_SetBitmapBits& Lib "gdi32" Alias "SetBitmapBits" (ByVal hBitmap&,
ByVal dwCount&, lpBits As Any)

' 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&)

' 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&)
```

```
' 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&)

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

' 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&)

' 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&)

' 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 Picture1 As Object
Var Shared Picture2 As Object
Var Shared Edit1 As Object
Var Shared Button1 As Object
Var SHared Bitmap As Object

Picture1.Attach GetDlgItem("Picture1")
Picture2.Attach GetDlgItem("Picture2")
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
BitmapObject Bitmap
Var Shared FileName As String
Var Shared bm As BITMAP

'=====================================
'= ClipBoardからPictureBoxへ
'=====================================
Declare Sub ImageFromClipBoard ()
Sub ImageFromClipBoard()
    Var hBitmap As Long
    Var phDC As Long
    Var mhDC As Long
    Var Ret As Long

    phDC = Api_GetDC(Picture2.GethWnd)

    Picture2.Cls

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

        '指定フォーマットのBITMAPデータを検索
        hBitmap = 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(hBitmap, Len(bm), bm)

        'Object選択
        Ret = Api_SelectObject(mhDC, hBitmap)

        '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, bm.bmWidth, bm.bmHeight, mhDC, 0, 0, SRCCOPY)

        '解放処理
        Ret = Api_ReleaseDC(Picture1.GethWnd, phDC)
        Ret = Api_DeleteDC(mhDC)
        Ret = Api_CloseClipboard()
    End If
End Sub

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

    'ファイルからBitmapをロード
    hBitmap = Api_LoadImage(0, FileName, IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE)

    If hBitmap = 0 Then
        A% = MessageBox("ロードできません！", "BitmapFileをEditBoxに" & CrLf & CrLf & "Drag &
Dropしてください！", 0, 2)
        Exit Sub
    End If

    '元の画像をPicture1に(F-Basic)
    Bitmap.LoadFile FileName
    Picture1.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject

    'オブジェクトを取得
    Ret = Api_GetObject(hBitmap, Len(bm), bm)

    Var Img(bm.bmWidthBytes - 1, bm.bmHeight - 1) As Byte

    'ビットマップのビットを取得し、バッファにコピー
    Ret = Api_GetBitmapBits(hBitmap, bm.bmWidthBytes * bm.bmHeight, Img(0, 0))

    '反転処理
    For y = 0 To bm.bmHeight - 1
        For x = 0 To bm.bmWidthBytes - 1
            Img(x, y) = Not Img(x, y)
        Next
    Next

    '反転されたビットマップ値をセット
    Ret = Api_SetBitmapBits(hBitmap, bm.bmWidthBytes * bm.bmHeight, Img(0, 0))

    'クリップボードにデータを設定
    Ret = Api_OpenClipboard(GethWnd)
    Ret = Api_EmptyClipboard
    Ret = Api_SetClipboardData(CF_BITMAP, hBitmap)
    Ret = Api_CloseClipboard

    'ClipboardからPicture2に画像を転送
    ImageFromClipBoard
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
```

画像を回転させる(Ⅰ)

画像を指定の角度に回転させます。
GetPixel 指定された座標のピクセルのRGB値を取得
SetPixel 指定した座標に点を配置する
GetDC 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

回転角度を指定して画像を回転させてみます。左から3～5番目のピクチャボックスには、それぞれ90°、180°、270°回転させています。
あまり大きな数値を入れるとピクチャボックスからはみ出ます‥


```
'=====================================
'= 画像を回転させる
'=    (PicRotate.bas)
'=====================================
#include "Windows.bi"

' 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&)

' 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 Picture(4) As Object
Var Shared Text1 As Object
Var Shared Edit1 As Object
Var Shared Button1 As Object
Var Shared Bitmap As Object

For i = 0 To 4
    Picture(i).Attach GetDlgItem("Picture" & Trim$(Str$(i + 1)))
Next
```

```
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
BitmapObject Bitmap

Var Shared hDC(4) As Long

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

    Bitmap.LoadFile "bike1.bmp"
    Picture(0).DrawBitmap Bitmap, 12, 14
    Bitmap.DeleteObject

    For i = 0 To 4
        hDC(i) = Api_GetDC(Picture(i).GethWnd)
    Next i
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var pai As single
    Var col As Long
    Var x As Long
    Var y As Long
    Var x1 As Long
    Var x2 As Long
    Var x3 As Long
    Var x4 As Long
    Var y1 As Long
    Var y2 As Long
    Var y3 As Long
    Var y4 As Long
    Var sin1 As double
    Var sin2 As double
    Var sin3 As double
    Var sin4 As double
    Var cos1 As double
    Var cos2 As double
    Var cos3 As double
    Var cos4 As double
    Var angle As Integer
    Var pWidth As Long
    Var pHeight As Long
    Var Ret As Long

    pai = 3.14159265
    angle = Val(Edit1.GetWindowtext)

    pWidth = Picture(0).GetWidth
    pHeight = Picture(0).GetHeight

    cos1 = Cos(angle * pai / 180)
    sin1 = Sin(angle * pai / 180)

    cos2 = Cos( 90 * pai / 180)
    sin2 = Sin( 90 * pai / 180)

    cos3 = Cos(180 * pai / 180)
    sin3 = Sin(180 * pai / 180)

    cos4 = Cos(270 * pai / 180)
    sin4 = Sin(270 * pai / 180)
```

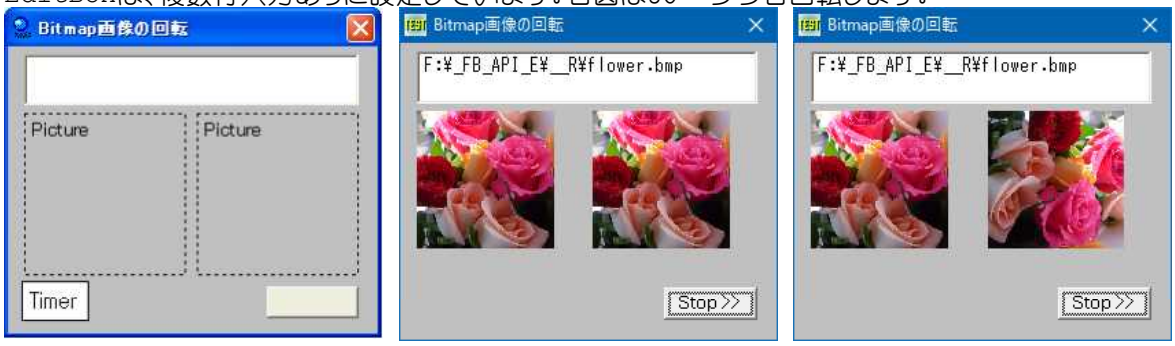
```
    For i = 1 To 4
        Picture(i).Cls
    Next i

    For x = -pWidth To pWidth
        For y = -pHeight To pHeight
            x1 = x * cos1 - y * sin1 + angle - 4
            y1 = x * sin1 + y * cos1 - 20

            x2 = x * cos2 - y * sin2 + pWidth
            y2 = x * sin2 + y * cos2

            x3 = x * cos3 - y * sin3 + pWidth
            y3 = x * sin3 + y * cos3 + pHeight

            x4 = x * cos4 - y * sin4
            y4 = x * sin4 + y * cos4 + pHeight

            '指定された座標のピクセルのRGB値を取得
            col = Api_GetPixel(hDC(0), x, y)

            '角度を変更して描画
            Ret = Api_SetPixel(hDC(1), x1, y1, col)
            Ret = Api_SetPixel(hDC(2), x2, y2, col)
            Ret = Api_SetPixel(hDC(3), x3, y3, col)
            Ret = Api_SetPixel(hDC(4), x4, y4, col)
        Next y
    Next x
End Sub

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

    For i = 0 To 4
        Ret = Api_ReleaseDC(Picture(i).GethWnd, hDC(i))
    Next i
End Sub

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

画像を回転させる(II)

Bitmap 画像を90度の角度で回転させます。
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
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
BitBlt ビットブロック転送を行う
GetObject オブジェクト取得
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
DeleteDC 指定されたデバイスコンテキストを削除
DeleteObject 論理オブジェクトを削除
GetDC 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

EditBoxにBmpファイルをドラッグ＆ドロップします。

EditBoxは、複数行入力ありに設定しています。右図は90°づつ右回転します。




```
'=====================================
'= 画像を回転させる(II)
'=    (RotateBitmap.bas)
'=====================================
#include "Windows.bi"

#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 LR_CREATEDIBSECTION &H2000          'デバイス独立ビットマップとしてロードする
#define SRCCOPY &HCC0020                    'そのまま転送
#define PI 3.14159

Type BITMAP
    bmType          As Long                 'ビットマップ タイプを指定(0)
    bmWidth         As Long                 'ビットマップの幅を指定
    bmHeight        As Long                 'ビットマップの高さを指定
    bmWidthBytes    As Long                 '1 走査線あたりのバイト数を指定
    bmPlanes        As Integer              'カラープレーンを指定(通常1)
    bmBitsPixel     As Integer              '1 ピクセルを定義するのに必要なビット数を指定
    bmBits          As Long                 'ビット データが格納されているの配列へのポインタを指定
End Type

' 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&)

' 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&)

' 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&)

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

' 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を削除
Declare Function Api_DeleteDC& Lib "gdi32" Alias "DeleteDC" (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&)

' 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 Timer1 As Object
Var Shared Picture1 As Object
Var Shared Picture2 As Object
Var Shared Edit1 As Object
Var Shared Button1 As Object

Timer1.Attach GetDlgItem("Timer1")
Picture1.Attach GetDlgItem("Picture1")
Picture2.Attach GetDlgItem("Picture2")
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var Shared sFileName As String
Var Shared hBitmap As Long
Var Shared lBMDC As Long
Var Shared bm As BITMAP
Var Shared Degrees As Long
Var Shared phDC1 As Long
Var Shared phDC2 As Long

'=====================================
'=
'=====================================
Declare Function Min(X1 As Long, Y1 As Long) As Long
Function Min(X1 As Long, Y1 As Long) As Long
    If X1 >= Y1 Then
        Min = Y1
    Else
        Min = X1
    End If
End Function

'=====================================
'=
'=====================================
Declare Function Max(X1 As Long, Y1 As Long) As Long
Function Max(X1 As Long, Y1 As Long) As Long
    If X1 >= Y1 Then
        Max = X1
    Else
        Max = Y1
    End If
End Function

'=====================================
'=
'=====================================
Declare Sub RotateBitmap(hBitmapDC As Long, lWidth As Long, lHeight As Long, lRadians As
Long)
Sub RotateBitmap(hBitmapDC As Long, lWidth As Long, lHeight As Long, lRadians As Long)
    Var hNewBitmapDC As Long
    Var hNewBitmap As Long
    Var lSine As Long
    Var lCosine As Long
    Var X1 As Long
    Var X2 As Long
    Var X3 As Long
    Var Y1 As Long
    Var Y2 As Long
    Var Y3 As Long
    Var lMinX As Long
    Var lMaxX As Long
    Var lMinY As Long
    Var lMaxY As Long
    Var lNewWidth As Long
    Var lNewHeight As Long
    Var i As Long
```

```
    Var j As Long
    Var lSourceX As Long
    Var lSourceY As Long
    Var Ret As Long

    hNewBitmapDC = Api_CreateCompatibleDC(hBitmapDC)

    lSine = Sin(lRadians)
    lCosine = Cos(lRadians)

    X1 = -lHeight * lSine
    Y1 = lHeight * lCosine
    X2 = lWidth * lCosine - lHeight * lSine
    Y2 = lHeight * lCosine + lWidth * lSine
    X3 = lWidth * lCosine
    Y3 = lWidth * lSine

    '新しいBitmapの最大/最小サイズ
    lMinX = Min(0, Min(X1, Min(X2, X3)))
    lMinY = Min(0, Min(Y1, Min(Y2, Y3)))
    lMaxX = Max(X1, Max(X2, X3))
    lMaxY = Max(Y1, Max(Y2, Y3))

    '新しいBitpamの幅/高さ
    lNewWidth = lMaxX - lMinX
    lNewHeight = lMaxY - lMinY

    '新しい幅/高さで回転するBitmapを作成
    hNewBitmap = Api_CreateCompatibleBitmap(hBitmapDC, lNewWidth, lNewHeight)

    'デバイスコンテキストのオブジェクトを選択
    Ret = Api_SelectObject(hNewBitmapDC, hNewBitmap)

    '各pixelを計算
    For i = 0 To lNewHeight
        For j = 0 To lNewWidth
            lSourceX = (j + lMinX) * lCosine + (i + lMinY) * lSine
            lSourceY = (i + lMinY) * lCosine - (j + lMinX) * lSine
            If (lSourceX >= 0) And (lSourceX <= lWidth) And (lSourceY >= 0) And (lSourceY <=
lHeight) Then
                Ret = Api_BitBlt(hNewBitmapDC, j, i, 1, 1, hBitmapDC, lSourceX, lSourceY,
SRCCOPY)
            End If
        Next j
    Next i

    '新しいBitmapの幅/高さをリセット
    lWidth = lNewWidth
    lHeight = lNewHeight

    '新しいBitmapにデバイスコンテキストを返す
    hBitmapDC = hNewBitmapDC

    'オブジェクトを削除
    Ret = Api_DeleteObject(hNewBitmap)
End Sub

'=======================================
'=
'=======================================
Declare Sub LoadBMPFromFile ()
Sub LoadBMPFromFile()

    '初期角度設定
    Degrees = 0
    sFileName = Edit1.GetWindowText
    Picture1.Cls
    Picture2.Cls
```

```
    'Bitmapをメモリにロード
    hBitmap = Api_LoadImage(0, sFileName, IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE Or
LR_CREATEDIBSECTION)

    '呼び出し失敗の場合終了
    If (hBitmap = 0) Then
        A% = MessageBox("Bitmap Load Error", "Error, Bitmapファイルではありません！", 0, 2)
        End
    End If

    'ロードしたBitmapのデバイスコンテキストを作成
    lBMDC = Api_CreateCompatibleDC(0)

    '失敗した場合プロシージャを抜ける
    If (lBMDC = 0) Then
        A% = MessageBox("Device Context Error", "Error, デバイスコンテキストを作成できません！",
0, 2)
        Exit Sub
    End If

    '作成されたデバイスコンテキストにBitmapをAttach
    Ret = Api_SelectObject(lBMDC, hBitmap)

    'イメージ情報を取得
    Ret = Api_GetObject(hBitmap, Len(bm), bm)

    'Pictureに描画
    Ret = Api_BitBlt(phDC1, 0, 0, bm.bmWidth, bm.bmHeight, lBMDC, 0, 0, SRCCOPY)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    phDC1 = Api_GetDC(Picture1.GethWnd)
    phDC2 = Api_GetDC(Picture2.GethWnd)

    'タイマーインターバルセット
    Timer1.SetInterval 100
    Timer1.Enable False
End Sub

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

    'ファイルからBitmapを読み込む
    LoadBMPFromFile

    If Timer1.IsEnabled = -1 Then
        Timer1.Enable 0
        Button1.SetWindowText "Start >>"
    Else
        Timer1.Enable -1
        Button1.SetWindowText "Stop >>"
        LoadBMPFromFile
    End If
End Sub

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    Var hRotatedBitmapDC As Long
    Var lWidth As Long
    Var lHeight As Long
```

```
    Var lRadians As Long

    Picture2.Cls

    'Bitmapイメージの幅/高さを設定
    lWidth = bm.bmWidth
    lHeight = bm.bmHeight

    '回転角度をラジアンに変換
    lRadians = PI * Degrees / 180

    'Picture1デバイスコンテキストを回転デバイスコンテキストに
    hRotatedBitmapDC = phDC1

    '新しい角度に回転させる
    RotateBitmap hRotatedBitmapDC, lWidth, lHeight, lRadians

    '回転イメージを転送
    Ret = Api_BitBlt(phDC2, 0, 0, lWidth, lHeight, hRotatedBitmapDC, 0, 0, SRCCOPY)

    'デバイスコンテキストを破棄
    Ret = Api_DeleteDC(hRotatedBitmapDC)

    '角度を90°増加。一巡したら0°に戻す
    Degrees = Degrees + 90

    If Degrees = 360 Then
        Degrees = 0
    End If
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)
    sFileName = GetDropFileName(DF, 0)
    Edit1.SetWindowText sFileName

    Button1.SetWindowText "Start >>"
End Sub

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

    Ret = Api_ReleaseDC(Picture1.GethWnd, phDC1)
    Ret = Api_ReleaseDC(Picture2.GethWnd, phDC2)
    End
End Sub

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

画像を回転させる(III)

画像を指定の角度に回転させます。
GetPixel 指定された座標のピクセルのRGB値を取得
SetPixel 指定した座標に点を配置する
GetDC 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


```
'=======================================
'= 画像を回転させる(III)
'=    (SetPixel2.bas)
'=======================================
#include "Windows.bi"

' 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&)

' 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 Edit1 As Object
Var Shared Picture1 As Object
Var Shared Picture2 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Picture1.Attach GetDlgItem("Picture1")
Picture2.Attach GetDlgItem("Picture2")
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Var Shared Bitmap As Object
BitmapObject Bitmap

Var Shared pi As Double
Var Shared hDc1 As Long
Var Shared hDc2 As Long

'=======================================
'=
'=======================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    pi = 3.14159265358979
    Bitmap.LoadFile "flower.bmp"
    Picture1.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject

    hDc1 = Api_GetDC(Picture1.GethWnd)
    hDc2 = Api_GetDC(Picture2.GethWnd)
```

```
End Sub

'====================================
'=
'====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var px As Integer
    Var py As Integer
    Var x As Integer
    Var y As Integer
    Var r As Integer
    Var s As Double
    Var a As Double
    Var Ret As Long

    On Error Goto *Er_Trap

    r = Val(Edit1.GetWindowText)
    s = -r * pi / 180

    px = Picture1.GetWidth * Cos(r * pi / 180) + Picture1.GetHeight * Sin(r * pi / 180)
    py = Picture1.GetHeight * Cos(r * pi / 180) + Picture1.GetWidth * Sin(r * pi / 180)

    Picture2.SetWindowSize px, py
    Picture2.Cls

    Var p(Picture1.GetWidth, Picture1.GetHeight) As Long

    For x = 0 To Picture1.GetWidth
        CallEvent
        For y = 0 To Picture1.GetHeight
            p(x, y) = Api_GetPixel(hDc1, x, y)
        Next
    Next

    a = Picture1.GetHeight * Sin(r * pi / 180)

    For x = -a To px
        CallEvent
        For y = 0 To py
           Ret = Api_SetPixel(hDc2, x + a, y, p(Int(x * Cos(s) - y * Sin(s)), Int(y * Cos(s) +
x * Sin(s))))
        Next
    Next
    Exit Sub

*Er_Trap
    Resume Next
End Sub

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

    Ret = Api_ReleaseDC(Picture1.GethWnd, hDc1)
    Ret = Api_ReleaseDC(Picture2.GethWnd, hDc2)
End Sub



'====================================
'=
'====================================
While 1
    WaitEvent
Wend
```

```
Stop
End
```

画像を上下･左右反転して転送

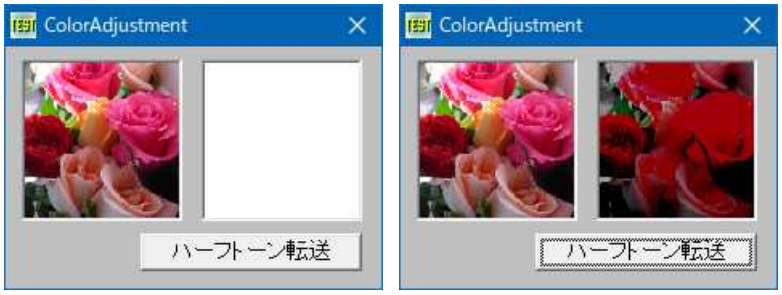
画像を上下･左右反転して転送します。
StretchBlt 拡縮をともなうグラフィックデバイス間のイメージを転送


```
'=====================================
'= 画像を上下･左右反転して転送
'=    (StretchBlt2.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                    'そのまま転送
#define SRCAND &H8800C6                     '転送先の画像とAND演算して転送
#define SRCPAINT &HEE0086                   '転送先の画像とOR演算して転送
#define SRCINVERT &H660046                  '転送先の画像とXOR演算して転送
#define NOTSRCCOPY &H330008                 '色を反転して転送

Var Shared Bitmap As Object
BitmapObject Bitmap
Var Shared Picture1 As Object
Var Shared Picture2 As Object
Var Shared Picture3 As Object
Var Shared Picture4 As Object

Picture1.Attach GetDlgItem("Picture1")
Picture2.Attach GetDlgItem("Picture2")
Picture3.Attach GetDlgItem("Picture3")
Picture4.Attach GetDlgItem("Picture4")

Var Shared hDC1 As Long
Var Shared hDC2 As Long
Var Shared hDC3 As Long
Var Shared hDC4 As Long



'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Bitmap.LoadFile "Bike1.bmp"
```

```
    Picture1.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject

    hDC1 = Api_GetDC(Picture1.GethWnd)
    hDC2 = Api_GetDC(Picture2.GethWnd)
    hDC3 = Api_GetDC(Picture3.GethWnd)
    hDC4 = Api_GetDC(Picture4.GethWnd)
End Sub

'=====================================
'=  通常拡大縮小
'=====================================
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, SRCCOPY)
    Ret = Api_ReleaseDC(Picture1.GethWnd, dDC1)
    Ret = Api_ReleaseDC(Picture2.GethWnd, dDC2)
End Sub

'=====================================
'=  左右反転拡大縮小
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Ret As Long

    Ret = Api_StretchBlt(hDC3, Picture3.GetWidth, 0, -Picture3.GetWidth,
Picture3.GetHeight, hDC1, 0, 0, Picture1.GetWidth,Picture1.GetHeight, SRCCOPY)
    Ret = Api_ReleaseDC(Picture1.GethWnd, dDC1)
    Ret = Api_ReleaseDC(Picture3.GethWnd, dDC3)
End Sub

'=====================================
'=  上下反転拡大縮小
'=====================================
Declare Sub Button3_on edecl ()
Sub Button3_on()
    Var Ret As Long

    Ret = Api_StretchBlt(hDC4, 0, Picture4.GetHeight, Picture4.GetWidth,
-Picture4.GetHeight, hDC1, 0, 0, Picture1.GetWidth, Picture1.GetHeight, SRCCOPY)
    Ret = Api_ReleaseDC(Picture1.GethWnd, dDC1)
    Ret = Api_ReleaseDC(Picture4.GethWnd, dDC4)
End Sub

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

画像をハーフトーンで転送

画像をハーフトーンで転送します。
StretchBlt ビットマップの拡大及び縮小
SetStretchBltMode ビットマップのストレッチングモードを設定
GetColorAdjustment デバイスコンテキストのカラー補正値を取得
SetColorAdjustment デバイスコンテキストのカラー補正値を設定
GetStretchBltMode 現在のストレッチングモードを取得




```
'=====================================
'= 画像をハーフトーンで転送
'= COLORADJUSTMENT構造体は、StretchBlt モードが HALFTONE のときに、
'= Windows の StretchBlt 関数と StretchDIBits 関数で使われるカラー調整値を定義
'=    (ColorAdjustment.bas)
'=====================================
#include "Windows.bi"

#define ILLUMINANT_A 1                      'タングステンランプ
#define HALFTONE 4                          'ハーフトーン
#define SRCCOPY &HCC0020                    'そのまま転送

Type COLORADJUSTMENT
    caSize            As Integer
    caFlags           As Integer
    caIlluminantIndex As Integer
    caRedGamma        As Integer
    caGreenGamma      As Integer
    caBlueGamma       As Integer
    caReferenceBlack  As Integer
    caReferenceWhite  As Integer
    caContrast        As Integer
    caBrightness      As Integer
    caColorfulness    As Integer
    caRedGreenTint    As Integer
End Type

' ビットマップの拡大及び縮小
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&)

' ビットマップのストレッチングモードを設定
Declare Function Api_SetStretchBltMode& Lib "gdi32" Alias "SetStretchBltMode" (ByVal
hDC&, ByVal nStretchMode&)

' デバイスコンテキストのカラー補正値を取得
Declare Function Api_GetColorAdjustment& Lib "gdi32" Alias "GetColorAdjustment" (ByVal
hDC&, lpca As COLORADJUSTMENT)

' デバイスコンテキストのカラー補正値を設定
Declare Function Api_SetColorAdjustment& Lib "gdi32" Alias "SetColorAdjustment" (ByVal
hDC&, lpca As COLORADJUSTMENT)

' 現在のストレッチングモードを取得
Declare Function Api_GetStretchBltMode& Lib "gdi32" Alias "GetStretchBltMode" (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 Picture1 As Object
Var Shared Picture2 As Object
Var Shared Bitmap As Object
BitmapObject Bitmap
```

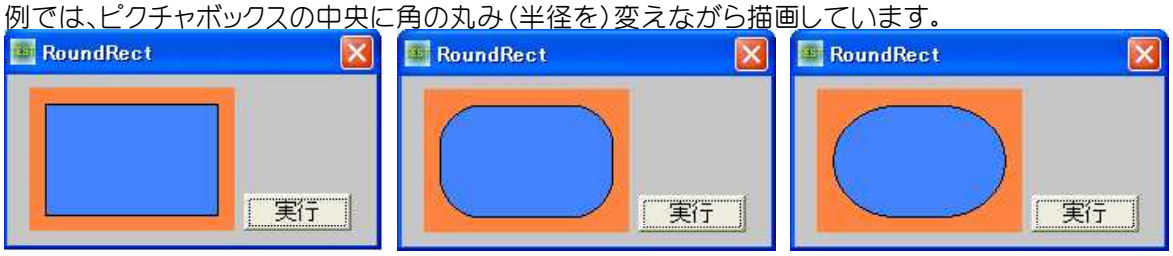
```
Picture1.Attach GetDlgItem("Picture1")
Picture2.Attach GetDlgItem("Picture2")

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Bitmap.LoadFile "Sample.bmp"
    Picture1.StretchBitmap Bitmap, 0, 0, Picture1.GetWidth, Picture1.GetHeight
'   Picture1.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteBitmap
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var CA As COLORADJUSTMENT
    Var hDC1 As Long
    Var hDC2 As Long
    Var p1Width As Long
    Var p1Height As Long
    Var Ret As Long

    CallEvent

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

    p1Width = Picture1.GetWidth
    p1Height = Picture1.GetHeight

    '現在のカラーを取得
    Ret = Api_GetColorAdjustment(hDC2, CA)

    'タイプの初期化
    CA.caSize = Len(CA)

    'ブライトネスを暗く設定
    CA.caBrightness = -100

    '新しい光源を設定
    CA.caIlluminantIndex = ILLUMINANT_A

    'ハーフトーンが設定されているかチェック
    If Api_GetStretchBltMode(hDC2) <> HALFTONE Then

        'ハーフトーンでなければハーフトーンに設定
       Ret = Api_SetStretchBltMode(HDC2, HALFTONE)
    End If

    'カラーアジャストメントのアップデート
    Ret = Api_SetColorAdjustment(hDC2, CA)

    'ピクチャ1からピクチャ2にコピー
    Ret = Api_StretchBlt(hDC2, 0, 0, p1Width, p1Height, hDC1, 0, 0, p1Width, p1Height,
SRCCOPY)

    '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
```

角の丸い矩形を描画(Ⅰ)

ピクチャボックスに角の丸い長方形を描画してみます。
RoundRect 角の丸い矩形を描画
GetDC 指定されたウィンドウのデバイスコンテキストを取得
ReleaseDC デバイスコンテキストを解放
CreateSolidBrush 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 オブジェクトを削除

例では、ピクチャボックスの中央に角の丸み(半径を)変えながら描画しています。



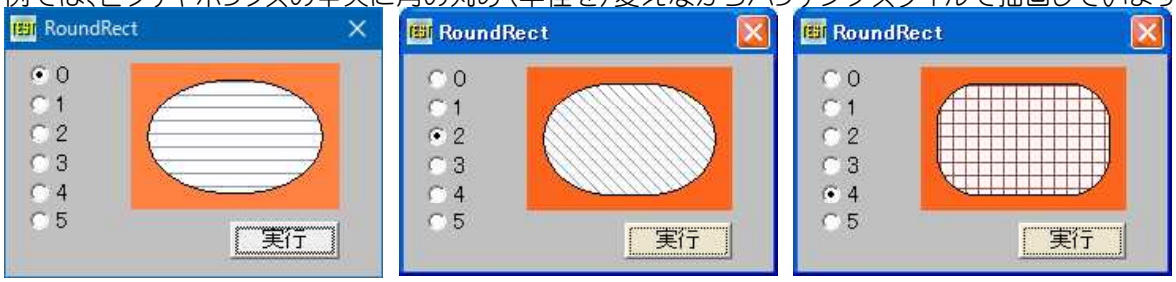

```
'=====================================
'= 角の丸い矩形を描画
'=    (RoundRect.bas)
'=====================================
#include "Windows.bi"

' 角の丸い矩形を描画
Declare Function Api_RoundRect& Lib "gdi32" Alias "RoundRect" (ByVal hDC&, ByVal
nLeftRect&, ByVal nTopRect&, ByVal nRightRect&, ByVal nBottomRect&, 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&)

' 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&)

' 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&)

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

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hDC As Long
    Var Radius As Integer
    Var hNewBrush As Long
    Var hOldBrush As Long
    Var Ret As Long
```

```
    hDC = Api_GetDC(Picture1.GethWnd)
    Picture1.SetBackColor RGB(255, 128, 64)
    hNewBrush = Api_CreateSolidBrush(RGB(64, 128, 255))
    Ret = Api_SelectObject(hDC, hNewBrush)

    For radius = 0 To 100 Step 5
        Picture1.cls
        Ret = Api_RoundRect(hDC, 10, 10, Picture1.GetWidth - 10 , Picture1.GetHeight - 10,
Radius, Radius)
        Wait 30
    Next

    Ret = Api_SelectObject(hDC, hOldBrush)
    Ret = Api_DeleteObject(nNewBrush)
    Ret = Api_ReleaseDC(Picture1.GethWnd, hDC)
End Sub

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

角の丸い矩形を描画(II)

ピクチャボックスに角の丸い長方形を描画してみます。
RoundRect 角の丸い矩形を描画
GetDC 指定されたウィンドウのデバイスコンテキストを取得
ReleaseDC デバイスコンテキストを解放
CreateBrushIndirect LOGBRUSH構造体を定義して論理ブラシを作成
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 オブジェクトを削除

例では、ピクチャボックスの中央に角の丸み(半径を)変えながらハッチングスタイルで描画しています。




```
'=====================================
'= 角の丸い矩形を描画(II)
'=    (RoundRect2.bas)
'=====================================
#include "Windows.bi"

Type LOGBRUSH
    lbStyle    As Long
    lbColor    As Long
    lbHatch    As Long
End Type

#define HS_BDIAGONAL 3                  '斜線(左上-右下)
#define HS_CROSS 4                      '水平と垂直クロスハッチ
#define HS_DIAGCROSS 5                  '45度のクロスハッチ
#define HS_FDIAGONAL 2                  '45度下向きのハッチ(左から右へ)
#define HS_HORIZONTAL 0                 '水平ハッチ
#define HS_VERTICAL 1                   '垂直ハッチ
#define BS_HATCHED 2                    'ハッチング(スタイルはlbHatchで指定)
```

```
' 角の丸い矩形を描画
Declare Function Api_RoundRect& Lib "gdi32" Alias "RoundRect" (ByVal hDC&, ByVal
nLeftRect&, ByVal nTopRect&, ByVal nRightRect&, ByVal nBottomRect&, 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&)

' 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&)

' LOGBRUSH構造体を定義して論理ブラシを作成
Declare Function Api_CreateBrushIndirect& Lib "gdi32" Alias "CreateBrushIndirect"
(lpLogBrush As LOGBRUSH)

' 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&)

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

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

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hDC As Long
    Var lb As LOGBRUSH
    Var Radius As Integer
    Var hNewBrush As Long
    Var hOldBrush As Long
    Var Ret As Long

    lb.lbStyle = BS_HATCHED
    lb.lbColor = RGB(Int(Rnd(1) * 255), Int(Rnd(1) * 255), Int(Rnd(1) * 255))
    lb.lbHatch = HS
    hDC = Api_GetDC(Picture1.GethWnd)
    Picture1.SetBackColor RGB(255, 128, 64)
    hNewBrush = Api_CreateBrushIndirect(lb)
    Ret = Api_SelectObject(hDC, hNewBrush)

    For radius = 0 To 100 Step 5
        Picture1.Cls
        Ret = Api_RoundRect(hDC, 10, 10, Picture1.GetWidth - 10 , Picture1.GetHeight - 10,
Radius, Radius)
        Wait 30
    Next

    Ret = Api_SelectObject(hDC, hOldBrush)
    Ret = Api_DeleteObject(nNewBrush)
    Ret = Api_ReleaseDC(Picture1.GethWnd, hDC)
End Sub

'=====================================
'=
'=====================================
While 1
```

```
    WaitEvent
Wend
Stop
End
```

角を丸めたフォーム(リージョン)

CreateRoundRectRgn 角を丸めた領域の設定
SetWindowRgn 指定領域をウィンドウ領域として設定
DeleteObject システムリソースの解放

メインフォームプロパティ
　　フレームの種類　→　境界線
　　タイトルバー　→　なし
図ではメインフォームと同サイズのビットマップを貼り付けています。(無くてもかまいません)
フォームをクリックする毎に、半径(R)が0～100まで10刻みで変化します。


```
'=====================================
'= リージョン(MAINFORMの角を丸く)
'=    (RoundRect.bas)
'=====================================
#include "Windows.bi"

' 角の丸い長方形のリージョンを作成
Declare Function Api_CreateRoundRectRgn& Lib "gdi32" Alias "CreateRoundRectRgn" (ByVal
nLeftRect&, ByVal nTopRect&, ByVal nRightRect&, ByVal nBottomRect&, ByVal
nWidthEllipse&, ByVal nHeightEllipse&)

' 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&)

' 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&)

Var Shared Bitmap As Object
Var Shared Button1 As Object

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

Var Shared R As Long                          '半径

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    On Error Goto *Er_Trap

    R = 0
    Bitmap.LoadFile "RoundRect.bmp"           '無くても可
    DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject
    Button1.SetWindowtext "半径→(" & Trim$(Str$(R))  & ")"
    Exit Sub

*Er_Trap
    Resume Next
```

```
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_On edecl ()
Sub Button1_On()
    Var Rgn As Long
    Var Ret As Long

    R = R + 10 : If R > 100 Then R = 0
    Rgn = Api_CreateRoundRectRgn(0, 0, GetWidth - 1, GetHeight - 1, R, R)
    Ret = Api_SetWindowRgn(GethWnd, Rgn, 1)
    Ret = Api_DeleteObject(Rgn)
    Button1.SetWindowtext "半径→(" & Trim$(Str$(R))  & ")"
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_On edecl ()
Sub Button2_On()
    End
End Sub

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

壁紙の設定と削除

壁紙の設定と削除を実行します。
SetDeskWallpaper デスクトップの壁紙を変更

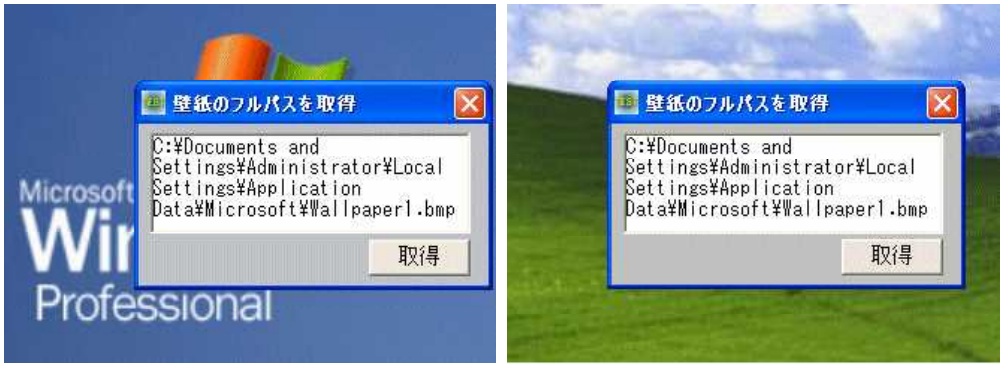
SetDeskWallpaperは、画面プロパティ→デスクトップ→表示位置での設定方法に依存します。(Windows2000では、設定できないようです)
最近は、SystemParametersInfo(SPI_SETDESKWALLPAPER, ...)に取って代わられたようです。


```
'=====================================
'= 壁紙の設定と削除
'=    (SetDeskWallpaper.bas)
'=====================================
#include "Windows.bi"

' デスクトップの壁紙を変更
Declare Function Api_SetDeskWallpaper& Lib "user32" Alias "SetDeskWallpaper" (ByVal
FileName$)
```

```
Var Shared Edit1 As Object
Var Shared Button1 As Object
Var Shared Button2 As Object
Var Shared Button3 As Object
Var Shared Picture1 As Object
Var Shared Bitmap As Object

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

Var Shared FileName As String

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    FileName = WinOpenDlg("ファイルのオープン", "*.bmp", "Bitmapﾌｧｲﾙ(*.bmp*)", 0)
    If FileName <> Chr$(&H1B) Then
        Edit1.SetWindowText FileName
        Bitmap.LoadFile FileName
        Picture1.StretchBitmap Bitmap, 0, 0, Picture1.GetWidth, Picture1.GetHeight
        Bitmap.DeleteObject
    End If
End Sub

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

    Ret = Api_SetDeskWallpaper(Edit1.GetWindowText)
End Sub

'=====================================
'= 壁紙削除
'=====================================
Declare Sub Button3_on edecl ()
Sub Button3_on()
    Var Ret As Long

    Ret = Api_SetDeskWallpaper("")
End Sub

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

壁紙のフルパスを取得

現在表示されている壁紙のフルパスを取得します。
SystemParametersInfo システム全体に関するパラメータを取得･設定
SPI_GETDESKWALLPAPER(115) 壁紙のフルパスを取得




```
'=====================================
'= 壁紙のフルパスを取得
'=   Windows2000以降
'=     (GetWallpaperPath.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_GETDESKWALLPAPER 115            '壁紙のフルパスを取得
#define MAX_PATH 260

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 WallpaperPath As String * MAX_PATH
    Var Ret As Long

    '壁紙のフルパスを取得
    Ret = Api_SystemParametersInfo(SPI_GETDESKWALLPAPER, Len(WallpaperPath),
WallpaperPath, 0)

    '壁紙のフルパスを表示
    Text1.SetWindowText Left$(WallpaperPath, InStr(WallpaperPath, Chr$(0)) - 1)
End Sub

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

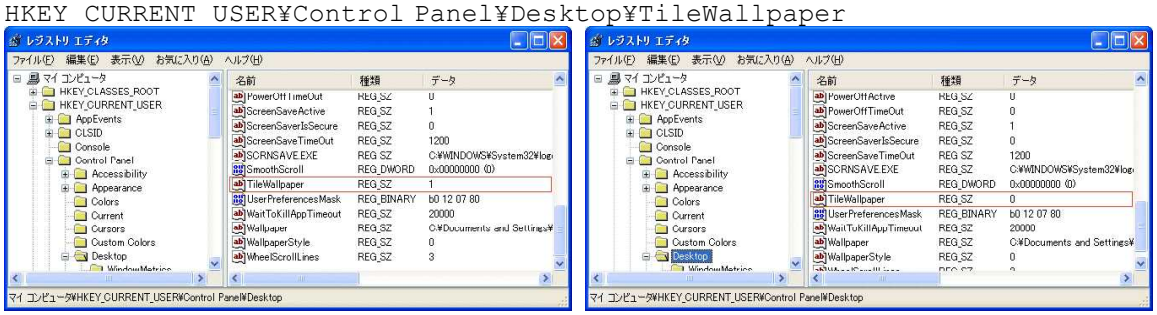
壁紙の変更

選択したビットマップ画像を画面いっぱいの壁紙にします。
SystemParametersInfo システム全体に関するパラメータのいずれかを取得または設定

レジストリに書き込んでいませんので、再起動した場合元に戻ります。
ファイルダイアログでBMPファイルを選択すると、その画像がPictureBoxに表示されます。
選択した画像がPictureBoxより大きい場合は拡大縮小で••(壁紙のイメージがつかめます)

壁紙の表示方法　レジストリに下記データを書き込む必要がありますが、今回は操作していません。確認だけしておきます。
壁紙を並べて表示する場合はデータを 1 に、または中央に壁紙を配置する場合は 0 値を設定します。

```
HKEY_CURRENT_USER¥Control Panel¥Desktop¥TileWallpaper
```



SystemParametersInfoについて　パラメータを指定する定数(UIACTION&)により下記のとおり書き方が変わります。
構造体を取得する場合(タスクバーを考慮してフォームを画面中央に/表示要素の寸法とシステム構成の設定を取得)

```
    declare Function API_SYSTEMPARAMETERSINFO& lib "user32" alias
"SystemParametersInfoA" (byval UIACTION&, byval UIPARAM&, PVPARAM as RECT, byval
FWININI&)
```

配列を取得する場合(壁紙を変更してみよう)

```
    declare Function API_SYSTEMPARAMETERSINFO& lib "user32" alias
"SystemParametersInfoA" (byval UIACTION&, byval UIPARAM&, PVPARAM as any, byval FWININI&)
```

長整数値を取得する場合

```
    declare Function API_SYSTEMPARAMETERSINFO& lib "user32" alias
"SystemParametersInfoA" (byval UIACTION&, byval UIPARAM&, PVPARAM&, byval FWININI&)
```

長整数値を設定する場合

```
    declare Function API_SYSTEMPARAMETERSINFO& lib "user32" alias
"SystemParametersInfoA" (byval UIACTION&, byval UIPARAM&, byval PVPARAM&, byval
FWININI&)
```

```
'=======================================
'= 壁紙の変更
'=    (WallPaper.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_SETDESKWALLPAPER 20             'デスクトップの壁紙を設定
#define SPIF_SENDWININICHANGE &H2           '全てのアプリケーションに通知して更新する
#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

Var Shared Button(3) As Object
Var Shared Text1 As Object
Var Shared Edit1 As Object
Var Shared Picture1 As Object
Var Shared Bitmap As Object
BitmapObject Bitmap
```

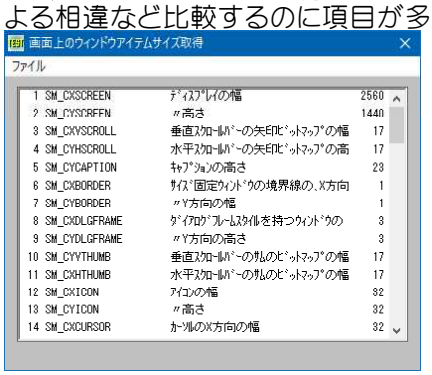
```
Picture1.Attach GetDlgItem("Picture1")
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
For i = 0 To 3
    Button(i).Attach GetDlgItem("Button" & Trim$(Str$(i + 1)))
    Button(i).SetFontSize 14
Next

Var Shared File As String
Var Shared FLG As byte

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Button(2).EnableWindow 0
    FLG = 0
End Sub

'=====================================
'=
'=====================================
Declare Sub BmpToPic edecl ()
Sub BmpToPic()
    If File <> "" Then
        Bitmap.LoadFile File
        Picture1.Cls

        If FLG = 0 Then
            Picture1.DrawBitmap Bitmap, 0, 0
        Else
            Picture1.StretchBitmap Bitmap,0 , 0, Picture1.GetWidth, Picture1.GetHeight
        End If
        Bitmap.DeleteObject
        Button(2).EnableWindow -1
    End If
End Sub

'=====================================
'=  ファイルオープンダイアログ
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    File = WinOpenDlg("ファイルのオープン", "*.bmp", "ﾋﾞｯﾄﾏｯﾌﾟﾌｧｲﾙ(*.bmp)", 0)

    If File <> Chr$(&H1B) Then
        Edit1.SetWindowText File
    End If
End Sub

'=====================================
'=  FLG切替(拡大縮小)
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    FLG = abs(FLG-1)

    BmpToPic
End Sub

'=====================================
'=  壁紙セット
'=====================================
Declare Sub Button3_on edecl ()
Sub Button3_on()
    Var Ret As Long

    Ret = Api_SystemParametersInfo(SPI_SETDESKWALLPAPER, 0, File, SPIF_UPDATEINIFILE Or
SPIF_SENDWININICHANGE)
```

```
End Sub

'=====================================
'= 壁紙消去
'=====================================
Declare Sub Button4_on edecl ()
Sub Button4_on()
    Var Ret As Long

    Ret = Api_SystemParametersInfo(SPI_SETDESKWALLPAPER, 0, Chr$(0), SPIF_UPDATEINIFILE
Or SPIF_SENDWININICHANGE)
End Sub

'=====================================
'=  EditBoxに変化があったら‥
'=====================================
Declare Sub Edit1_Change edecl ()
Sub Edit1_Change()
    File = GetDlgItemText("Edit1")

    BmpToPic
End Sub

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

画面上のウィンドウアイテムサイズ取得

画面およびフォーム上の各サイズを取得します。
SystemParametersInfo システムパラメータ情報の取得
GetSystemMetrics 表示要素寸法とシステム構成等の取得
SendMessage メッセージの送信

テストプログラムでは、定数および取得した値をリスト表示します。リスト内表示データは印刷することができます。OSによる相違など比較するのに項目が多すぎるので印刷するようにしました。(ファイル→印刷)(ファイル→終了)




```
'=====================================
'= 画面上のウィンドウアイテムサイズ取得
'=    (GetSystemMetrics.bas)
'=====================================
#include "Windows.bi"

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

```
' 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 RECT, ByVal fWinIni&)

' 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&)

' 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_GETITEMHEIGHT &H1A1              'リストボックス内の項目の高さを取得する
#define LB_SETITEMHEIGHT &H1A0              'リストボックス項目の高さを設定
#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 List1 As Object
Var Shared Bitmap As Object
Var Shared Prt As Object
Var Shared Prm As PRINTPARAM
Var Shared SM$(79,2) As String

List1.Attach GetDlgItem("List1") : List1.SetFontSize 12
BitmapObject Bitmap
PrinterObject Prt

'=====================================
'= ディスプレイ上のウィンドウアイテムのサイズを取得する
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    on error goto *Er_Trap

    Var Flg As byte
    Var rc As RECT
    Var Ret As Long
    Var ListHeight As Long
    Var nIndex As Long

    '------------------------------
    '二重起動禁止
    '------------------------------
    If PrevInstance Then
        A% = MessageBox(GetWindowText, "このプログラムは実行中です。" , 0 , 0 ) : End
    End If

    '------------------------------
    'ワークエリア取得
    '------------------------------
    Ret = Api_SystemParametersInfo(SPI_GETWORKAREA, 0, rc, 0)

    '------------------------------
    'リスト行間隔設定
    '------------------------------
    ListHeight = 18
    Ret = Api_SendMessage&(List1.GethWnd, LB_SETITEMHEIGHT, 0, ByVal ListHeight)

    '------------------------------
    '
    '------------------------------
    For i = 0 To 79
        For j = 0 To 2
            Read SM$(i, j)
        Next
    Next

    data SM_CXSCREEN          , 0,ディスプレイの幅
    data SM_CYSCREEN          , 1,  〃 高さ
    data SM_CXVSCROLL         , 2,垂直スクロールバーの矢印ビットマップの幅
    data SM_CYHSCROLL         , 3,水平スクロールバーの矢印ビットマップの高さ
```

```
data SM_CYCAPTION        , 4,キャプションの高さ
data SM_CXBORDER         , 5,サイズ固定ウィンドウの境界線の、X方向の幅
data SM_CYBORDER         , 6,  〃Y方向の幅
data SM_CXDLGFRAME       , 7,ダイアログフレームスタイルを持つウィンドウの枠線のX方向の幅
data SM_CYDLGFRAME       , 8,  〃Y方向の高さ
data SM_CYVTHUMB         , 9,垂直スクロールバーのサムのビットマップの幅
data SM_CXHTHUMB         ,10,水平スクロールバーのサムのビットマップの幅
data SM_CXICON           ,11,アイコンの幅
data SM_CYICON           ,12,  〃高さ
data SM_CXCURSOR         ,13,カーソルのX方向の幅
data SM_CYCURSOR         ,14,  〃Y方向の高さ
data SM_CYMENU           ,15,メニューバーの行の高さ
data SM_CXFULLSCREEN     ,16,最大化したときのクライアント領域の幅
data SM_CYFULLSCREEN     ,17,  〃高さ
data SM_CYKANJIWINDOW    ,18,漢字ウィンドウの高さ
data SM_MOUSEPRESENT     ,19,マウスがあるとき1
data SM_CYVSCROLL        ,20,垂直スクロールバーの矢印ビットマップの高さ
data SM_CXHSCROLL        ,21,水平スクロールバーの矢印ビットマップのX方向の幅
data SM_DEBUG            ,22,USER.EXEがデバッグ版のとき1
data SM_SWAPBUTTON       ,23,左右のマウスボタンの機能を切り替えているとき1
data SM_RESERVED1        ,24,予備1
data SM_RESERVED2        ,25,予備2
data SM_RESERVED3        ,26,予備3
data SM_RESERVED4        ,27,予備4
data SM_CXMIN            ,28,ウィンドウの最小幅
data SM_CYMIN            ,29,  〃高さ
data SM_CXSIZE           ,30,タイトルバー内のビットマップの幅
data SM_CYSIZE           ,31,  〃高さ
data SM_CXFRAME          ,32,サイズ可変ウィンドウの境界線の、X方向の幅
data SM_CYFRAME          ,33,  〃Y方向の幅
data SM_CXMINTRACK       ,34,ウィンドウの可能な最小幅
data SM_CYMINTRACK       ,35,  〃高さ
data SM_CXDOUBLECLK      ,36,ダブルクリックと見なせッル前後のクリックのX方向の幅
data SM_CYDOUBLECLK      ,37,  〃Y方向の幅
data SM_CXICONSPACING    ,38,アイコンを配置するための矩形の幅
data SM_CYICONSPACING    ,39,  〃高さ
data SM_MENUDROPALIGNMENT ,40,サブメニューが左側揃えのとき0
data SM_PENWINDOWS       ,41,ペンウィンドウのとき1
data SM_DBCSENABLED      ,42,2バイト文字をサポートしているとき1
data SM_CMOUSEBUTTONS    ,43,マウスボタンの数(マウスがないとき0)
data SM_CXFIXEDFRAME     , 7,SM_CXDLGFRAMEダイアログフレームスタイルを持つウィンドウの枠線のX
                         方向の幅
data SM_CYFIXEDFRAME     , 8,SM_CYDLGFRAME  〃Y方向の高さ
data SM_CXSIZEFRAME      ,32,SM_CXFRAMEサイズ可変ウィンドウの境界線の、X方向の幅
data SM_CYSIZEFRAME      ,33,SM_CYFRAME  〃Y方向の高さ
data SM_SECURE           ,44,セキュリティが存在するかどうかを示す値
data SM_CXEDGE           ,45,3D表示ためのX方向の幅
data SM_CYEDGE           ,46,  〃Y方向の幅
data SM_CXMINSPACING     ,47,最小化したウィンドウのグリッドセルの幅
data SM_CYMINSPACING     ,48,  〃高さ
data SM_CXSMICON         ,49,小さなキャプション内のビットマップの幅
data SM_CYSMICON         ,50,  〃高さ
data SM_CYSMCAPTION      ,51,小さいキャプションの高さ
data SM_CXSMSIZE         ,52,タイトルバー内の小さいボタンの幅
data SM_CYSMSIZE         ,53,  〃高さ
data SM_CXMENUSIZE       ,54,メニューバーボタンの幅
data SM_CYMENUSIZE       ,55,  〃高さ
data SM_ARRANGE          ,56,最小化ウィンドウの配置方法を示す値
data SM_CXMINIMIZED      ,57,ウィンドウの可能な最小幅
data SM_CYMINIMIZED      ,58,  〃高さ
data SM_CXMAXTRACK       ,59,サイズ変更可能なウィンドウの最大幅
data SM_CYMAXTRACK       ,60,  〃高さ
data SM_CXMAXIMIZED      ,61,ディスプレイの最大幅
data SM_CYMAXIMIZED      ,62,  〃高さ
data SM_NETWORK          ,63,ネットワークになっているとき
data SM_CLEANBOOT        ,67,Windowsを起動した方法
data SM_CXDRAG           ,68,ドラッグを検出する矩形の幅
data SM_CYDRAG           ,69,  〃Y高さ
data SM_SHOWSOUNDS       ,70,サウンド解説を使うが有効かどうかを判定
data SM_CXMENUCHECK      ,71,メニューボタンのチェックマークの幅
```

```
    data SM_CYMENUCHECK      ,72,  〃高さ
    data SM_SLOWMACHINE      ,73,処理速度の遅いマシンのとき1
    data SM_MIDEASTENABLED   ,74,アラビア語、ヘブライ語をサポートするとき1
    data SM_XVIRTUALSCREEN   ,76,仮想スクリーンの左座標
    data SM_YVIRTUALSCREEN   ,77,  〃上座標
    data SM_CXVIRTUALSCREEN  ,78,仮想スクリーンの幅
    data SM_CYVIRTUALSCREEN  ,79,  〃高さ

    '------------------------------
    'リスト表示
    '------------------------------
    List1.ResetContent
    For i = 0 To 79
        ZD1$ = Format$(i + 1," ## ")
        ZD2$ = Format$(Trim$(SM$(i, 0)),"&                & ")
        ZD3$ = Format$(Kacnv$(SM$(i, 2)),"&                          & ")
        ZD4$ = Format$(Api_GetSystemMetrics(Val(SM$(i, 1))),"  ####")
        List1.AddString ZD1$ & ZD2$ & ZD3$ & ZD4$
    Next
    List1.AddString String$(71, "-")

    '------------------------------
    'ワークエリア･タスクバー
    '------------------------------
    ZD1$ = "ワークエリア ("
    ZD2$ = Str$(rc.Left) & "," & Str$(rc.Top) & ")-("
    ZD3$ = Str$(rc.Right - 1) & "," & Str$(rc.Bottom - 1) & ")"
    List1.AddString ZD1$ & ZD2$ & ZD3$

    ZD1$ = "タスクバーの高さ "
    ZD2$ = Format$(Api_GetSystemMetrics(1) - rc.Bottom,"  ####")
    List1.AddString ZD1$ & ZD2$
    Exit Sub

    *Er_Trap
        Flg = 1 : Resume Next
End Sub

'=====================================
'= 印刷
'=====================================
Declare Sub mnuPrintOut_On edecl ()
Sub mnuPrintOut_On()
    Prt.SetupPrinterMode "SetupPrinter:", 13, 1
    If Prt.PrintDlg(Prm) <> 0 Then
        Prt.SetMapMode 2
        Prt.StartDoc "Sample"
        Prt.StartPage
        Prt.SetFontName "MS ゴシック"
        Prt.SetFontSize 11
        Page = 1
        Gyo = 0
        GoSub *Prt_Style
        For i = 0 To 79
            Gyo = Gyo + 1
            ZD1$ = Format$(i + 1," ## ")
            ZD2$ = Format$(Trim$(SM$(i, 0)),"&                & ")
            ZD3$ = Format$(Kacnv$(SM$(i, 2)),"&                          & ")
            ZD4$ = Format$(Api_GetSystemMetrics(Val(SM$(i, 1))),"  ####")
            Prt.Symbol(150, (Gyo - 1) * 45 + 350), ZD1$ & ZD2$ & ZD3$ & ZD4$, 1, 1
            If Gyo = 40 and i < 79 Then GoSub *Page_Change
        Next

        Prt.EndPage
        Prt.EndDoc
        Prt.ClosePrinter
        Exit Sub
    Else
        Exit Sub
    End If
```

```
    *Prt_Style
        Prt.Symbol(630, 200), "ウィンドウズアイテムサイズ取得", 1, 1
        Prt.Symbol(850, 2300), "- " & Trim$(Str$(Page)) & " -", 1, 1
        Return

    *Page_Change
        Prt.EndPage
        Prt.EndDoc

        Prt.StartDoc "Sample"
        Prt.StartPage
        Page = Page + 1
        Gyo = 0
        goSub *Prt_Style
        Return
End Sub

'=====================================
'=  終了
'=====================================
Declare Sub MainForm_QueryClose edecl (Cancel%, ByVal Mode%)
Sub MainForm_QueryClose(Cancel%, ByVal Mode%)
    Cancel% = MessageBox(GetWindowText, "終了しますか？", 1, 1)
    If Cancel% = 0 Then End
End Sub

Declare Sub mnuExit_On edecl ()
Sub mnuExit_On()
    Ret = MessageBox(GetWindowText, "終了しますか？", 1, 1)
    If Ret = 0 Then End
End Sub

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

画面の色数を取得

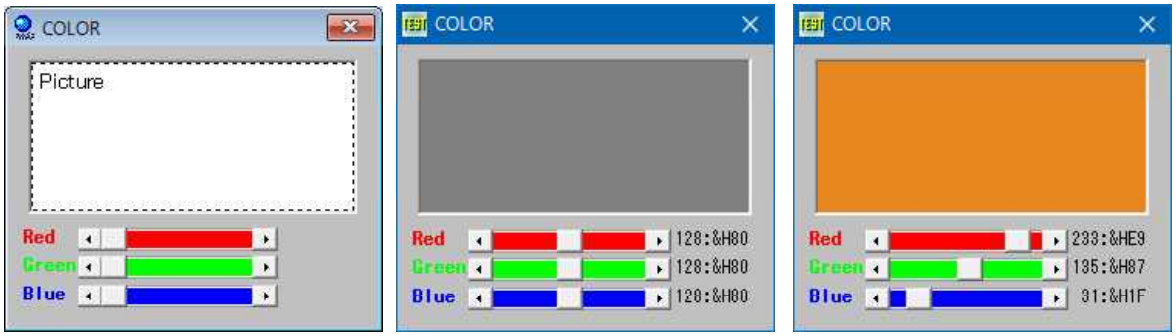
画面の色数を取得します。
GetDeviceCaps デバイス固有の情報を取得
GetDC デバイスコンテキストのハンドルを取得
ReleaseDC デバイスコンテキストを解放


```
'=====================================
'= 画面の色数を取得
'=    (GetDeviceCaps2.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&)

#define BITSPIXEL 12                    'ピクセルあたりのカラービットの数（プレーンごと）
#define PLANES 14                       'カラープレーン数

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 ColorDeepness() As String
Function ColorDeepness() As String
    Var hDC As Long
    Var cPixels As Long
    Var Txt As String
    Var Ret As Long

    hDC = Api_GetDC(0)
    cPixels = Api_GetDeviceCaps(hDC, BITSPIXEL) * Api_GetDeviceCaps(hDC, PLANES)
    Ret = Api_ReleaseDC(0, hDC)

    Select Case cPixels
        Case 1
            Txt = "2 色"
        Case 4
            Txt = "16 色"
        Case 8
            Txt = "256 色"
        Case 16
            Txt = "HighColor: " & Str$(2 ^ cPixels) & " 色"
        Case 32
            Txt = "TrueColor: " & Str$(2 ^ cPixels) & " 色"
    End Select

    ColorDeepness = Txt
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Text1.SetWindowText ColorDeepness
End Sub

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

カラー設定とRGB値

RGBをスクロールバーで設定し、その色を表示させます。




```
'=====================================
'= カラー設定とRGB値
'=    (COLOR.bas)
'=====================================
#include "Windows.bi"

Var Shared RScl As Object
Var Shared GScl As Object
Var Shared BScl As Object
Var Shared Pic As Object
Var Shared Text(5) As Object

'=====================================
'=
'=====================================
Declare Sub Scroll_Change edecl ()
Sub Scroll_Change()
    Pic.SetBackColor RGB( RScl.GetScrollPos, GScl.GetScrollPos, BScl.GetScrollPos)
    Pic.Cls
    Text(3).SetWindowText Format$(RScl.GetScrollPos, "###:") & "&&H" &
Hex$(RScl.GetScrollPos)
    Text(4).SetWindowText Format$(GScl.GetScrollPos, "###:") & "&&H" &
Hex$(GScl.GetScrollPos)
    Text(5).SetWindowText Format$(BScl.GetScrollPos, "###:") & "&&H" &
Hex$(BScl.GetScrollPos)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    RScl.Attach GetDlgItem("RScl")
    RScl.SETScrollPos 128
    GScl.Attach GetDlgItem("GScl")
    GScl.SETScrollPos 128
    BScl.Attach GetDlgItem("BScl")
    BScl.SETScrollPos 128

    Pic.Attach GetDlgItem("Picture1")
    For i = 0 To 5
        Text(i).Attach GetDlgItem("Text" & trim$(str$(i+1)))
        Text(i).SetFontSize 12
    Next

    Scroll_Change
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_QueryClose edecl (Cancel%, Mode%)
Sub MainForm_QueryClose(Cancel%, Mode%)
    End
End Sub
```

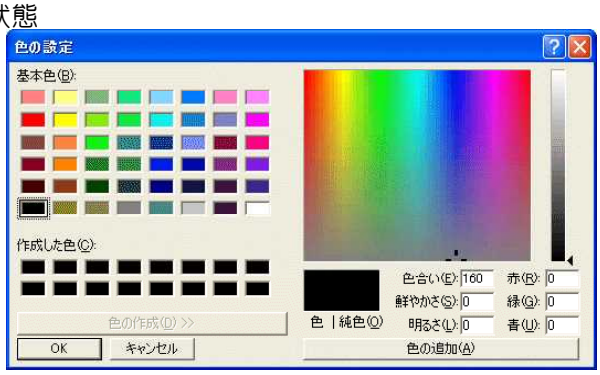
```
'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

カラー選択ダイアログを開く

ChooseColor カラー選択ダイアログを開きます。

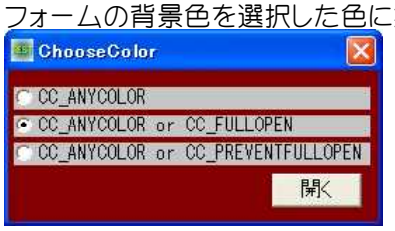
基本色を全て表示で開く

「色の作成」コントロールを開いた状態



「色の作成」コントロールの使用を禁止

フォームの背景色を選択した色に塗ります。




```
'=====================================
'= カラー選択ダイアログを開く
'=    (ChooseColor.bas)
'=====================================
```

```
#include "Windows.bi"

Type CHOOSECOLORSTRUCT
    lStructSize       As Long
    hwndOwner         As Long
    hInstance         As Long
    rgbResult         As Long
    lpCustColors      As Long
    flags             As Long
    lCustData         As Long
    lpfnHook          As Long
    lpTemplateName    As Long
End Type

' カラー選択ダイアログボックスを開く
Declare Function Api_ChooseColor& Lib "comdlg32" Alias "ChooseColorA" (lpcc As
CHOOSECOLORSTRUCT)

#define CC_RGBINIT           &H1          'crDef で指定された色を初期選択として使用
#define CC_FULLOPEN          &H2          '「色の作成」コントロールを開いた状態
#define CC_PREVENTFULLOPEN   &H4          '「色の作成」コントロールの使用を禁止
#define CC_SHOWHELP          &H8          '「ヘルプ」ボタンを表示
#define CC_ENABLEHOOK        &H10
#define CC_ENABLETEMPLATE    &H20         'nInstanceメンバとlpTemplateNameメンバで指定され
                                          たダイアログボックステンプレートを使ってダイアログを作成
#define CC_ENABLETEMPLATEHANDLE &H40
#define CC_SOLIDCOLOR        &H80         '基本色のうち、純色だけを表示
#define CC_ANYCOLOR          &H100        '利用可能な基本色をすべて表示

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

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

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

    cc.lStructSize = Len(CC)
    cc.hwndOwner = GethWnd
    If Radio(0).GetCheck = 1 Then cc.flags = CC_ANYCOLOR
    If Radio(1).GetCheck = 1 Then cc.flags = CC_ANYCOLOR Or CC_FULLOPEN
    If Radio(2).GetCheck = 1 Then cc.flags = CC_ANYCOLOR Or CC_PREVENTFULLOPEN
    cc.rgbResult = RGB(192, 192, 192)
    cc.lpCustColors = Varadr(BDF(0))

    Ret = Api_ChooseColor(cc)

    If Ret <> 0 Then
        BackColor = cc.rgbResult
        SetBackColor BackColor
        Cls
    End If
End Sub

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

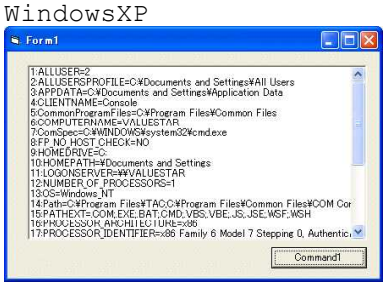
環境変数の取得(Ｉ)

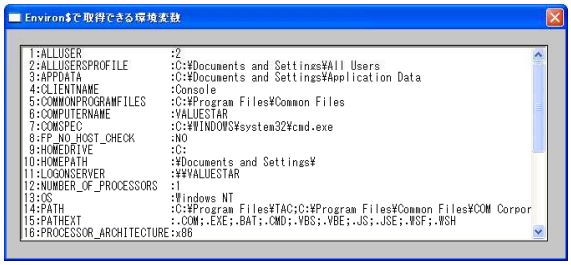
環境変数文字列は、どういうものがあるか判らなかったので、VisualBasicにて確認しdataとしています。
文字通り環境により変わります。

VisualBasicでの取得コード

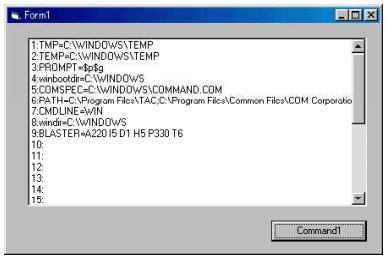
```
Private Sub Command1_Click()
    Dim i As Long
    For i = 1 To 29
        List1.AddItem i & Environ(i)
    Next
End Sub
```

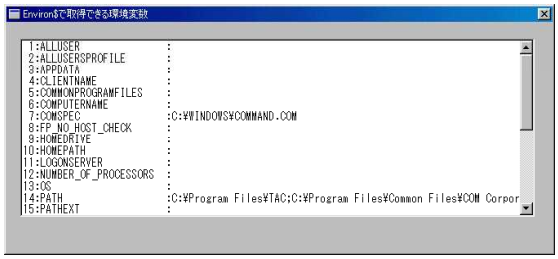
WindowsXP



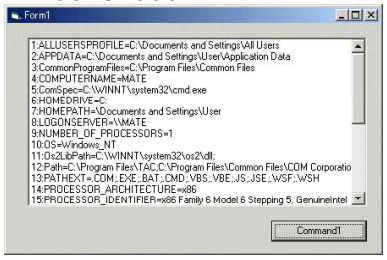


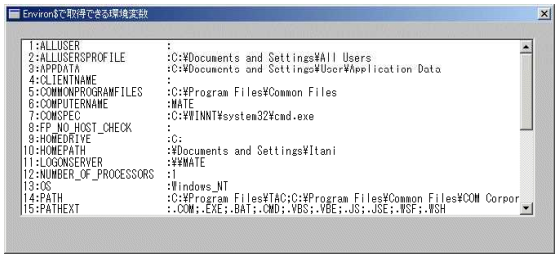
Windows98





Windows2000



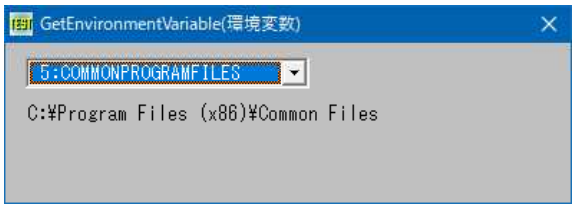



```
'=======================================
'=  環境変数取得(ENVIRON$("環境変数文字列")
'=======================================
#include "Windows.bi"

Var Shared MAINFORM As Object
Var Shared LIST1 As Object

MAINFORM.ATTACH GETHWND
LIST1.ATTACH GETDLGITEM("LIST1")
LIST1.SETFONTNAME "MS ゴシック"
LIST1.SETFONTSIZE 14

'=======================================
'=
'=======================================
Declare Sub MAINFORM_START edecl ()
Sub MAINFORM_START()
    MAINFORM.SETWINDOWTEXT "Environ$で取得できる環境変数"
    Var KH$(29) As String
    For I = 1 To 29
        read KH$(I)
    Next

    LIST1.RESETCONTENT
    For I = 1 To 29
```

```
        LIST1.ADDSTRING Format$(I,"##:") & Format$(KH$(I),"&                    &:") &
environ$(KH$(I))
    Next

    '--------------------
    data ALLUSER
    data ALLUSERSPROFILE
    data APPDATA
    data CLIENTNAME
    data CommonProgramFiles
    data COMPUTERNAME
    data ComSpec
    data FP_NO_HOST_CHECK
    data HOMEDRIVE
    data HOMEPATH
    data LOGONSERVER
    data NUMBER_OF_PROCESSORS
    data OS
    data Path
    data PATHEXT
    data PROCESSOR_ARCHITECTURE
    data PROCESSOR_IDENTIFIER
    data PROCESSOR_LEVEL
    data PROCESSOR_REVISION
    data ProgramFiles
    data SESSIONNAME
    data SystemDrive
    data SystemRoot
    data TEMP
    data TMP
    data USERDOMAIN
    data USERNAME
    data USERPROFILE
    data WINDIR
End Sub

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

環境変数の取得(II)

環境変数を取得します。
GetEnvironmentVariable 環境変数取得



F-BASICでのenviron$関数と同じです。

```
'=======================================
'= 環境変数取得
'=    (GetEnvironmentVariable.bas)
'=======================================
#include "Windows.bi"

' 環境変数取得
Declare Function Api_GetEnvironmentVariable& Lib "kernel32" Alias
"GetEnvironmentVariableA" (ByVal lpszName$, ByVal lpszValue$, ByVal cchValue&)
```

```
Var Shared Text1 As Object
Var Shared Combo1 As Object

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

Var Shared kh(29) As String

'=====================================
'=
'=====================================
Declare Function GetEnv(sName As String) As String
Function GetEnv(sName As String) As String
    Var Buffer As String
    Var Size As Long
    Var Ret As Long

    GetEnv = ""
    Buffer = String$(512, Chr$(0))
    Size = Api_GetEnvironmentVariable(sName, Buffer, 512)

    If Size = 0 Then
    Else If Size <= 512 Then
        GetEnv = Left$(Buffer, InStr(Buffer, Chr$(0)) - 1)
    Else
        Buffer = String$(Size, Chr$(0))
        Ret = Api_GetEnvironmentVariable(sName, Buffer, Size)
        If Ret = Size - 1 Then
            GetEnv = Left$(Buffer, InStr(Buffer, Chr$(0)) - 1)
        End If
    End If
End Function

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub mainForm_Start()
    Combo1.ResetContent

    For i = 1 To 29
        Read kh(i)
        Combo1.AddString Format$(i,"##:") & kh(i)
    Next

    '--------------------
    data ALLUSER
    data ALLUSERSPROFILE
    data APPDATA
    data CLIENTNAME
    data CommonProgramFiles
    data COMPUTERNAME
    data ComSpec
    data FP_NO_HOST_CHECK
    data HOMEDRIVE
    data HOMEPATH
    data LOGONSERVER
    data NUMBER_OF_PROCESSORS
    data OS
    data Path
    data PATHEXT
    data PROCESSOR_ARCHITECTURE
    data PROCESSOR_IDENTIFIER
    data PROCESSOR_LEVEL
    data PROCESSOR_REVISION
    data ProgramFiles
    data SESSIONNAME
    data SystemDrive
    data SystemRoot
```

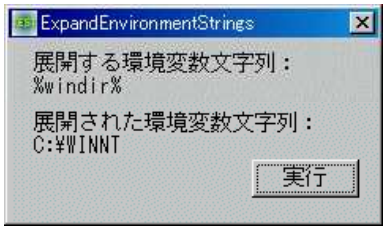
```
    data TEMP
    data TMP
    data USERDOMAIN
    data USERNAME
    data USERPROFILE
    data WINDIR
End Sub

'========================================
'=
'========================================
Declare Sub Combo1_Change edecl ()
Sub Combo1_Change()
    Var index As Long

    index = Combo1.GetCursel + 1
    Text1.SetWindowText GetEnv(kh(index))
End Sub

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

環境変数文字列を展開

ExpandEnvironmentStrings 環境変数文字列を展開し、その文字列を定義された値に置き換える

WindowsXP

Windows2000




```
'========================================
'= 環境変数文字列を展開
'=    (ExpandEnvironmentStrings.bas)
'========================================
#include "Windows.bi"

' 環境変数文字列を展開し、その文字列を定義された値に置き換える
Declare Function Api_ExpandEnvironmentStrings& Lib "kernel32" Alias
"ExpandEnvironmentStringsA" (ByVal lpSrc$, ByVal lpDst$, ByVal nSize&)

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 EnvSrc As String
    Var EnvDest As String * 256
    Var Ret As Long

    '展開する環境変数文字列を設定
    EnvSrc = "%windir%"
    Text1.SetWindowText "展開する環境変数文字列:" & Chr$(13, 10) & EnvSrc

    '環境変数文字列を展開
    Ret = Api_ExpandEnvironmentStrings(EnvSrc, EnvDest, Len(EnvDest))

    '環境変数文字列を表示
    Text2.SetWindowText "展開された環境変数文字列:" & Chr$(13, 10) & Left$(EnvDest,
InStr(EnvDest, Chr$(0)) - 1)
End Sub

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

管理者権限の有無を判断

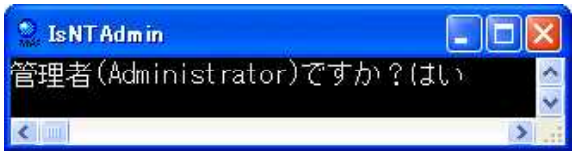
IsUserAnAdmin 管理者権限の有無を取得 (Windows2K以降)

```
'=====================================
'= 管理者権限の有無を判断
'=====================================

' 管理者権限があるかどうかを判断
Declare Function Api_IsUserAnAdmin& Lib "shell32" Alias "IsUserAnAdmin" ()

Var Ret As Long

Ret = Api_IsUserAnAdmin()

If Ret = 1 Then
    Print "管理者権限あり！"
Else
    Print "管理者権限なし！"
End If

Stop
End
```

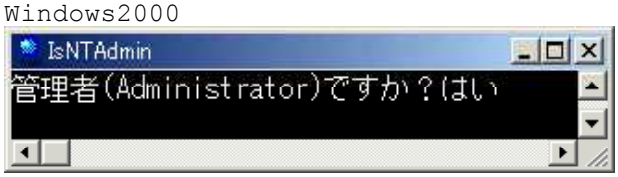
管理者権限のチェック

現在のユーザがWindows2000、WindowsXPにおける管理者権限をチェックします。
IsNTAdmin 管理者権限チェック
権限がある場合「0」以外を返します。

```
WindowsXP
```



Windows2000



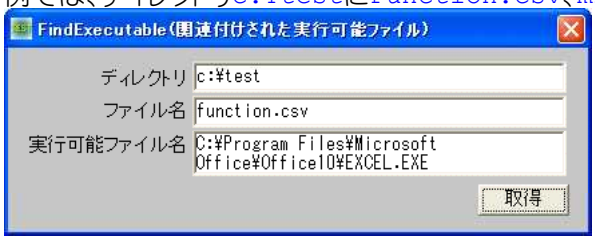
Windows98で実行した例

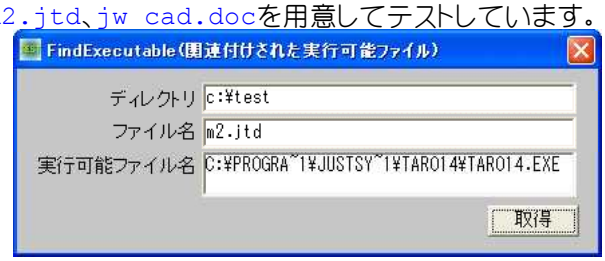

```
'=======================================
'= 管理者権限のチェック
'=    (IsNTAdmin.bas)
'=======================================

' 管理者権限チェック
Declare Function Api_IsNTAdmin& Lib "advpack" Alias "IsNTAdmin" (ByVal dwReserved&,
ByVal lpdwReserved&)

Var Ret As Long

Ret = Api_IsNTAdmin(ByVal 0, ByVal 0)

If Ret Then
    Print "管理者(Administrator)ですか？" & "はい"
Else
    Print "管理者(Administrator)ですか？" & "いいえ"
End If
Stop
End
```

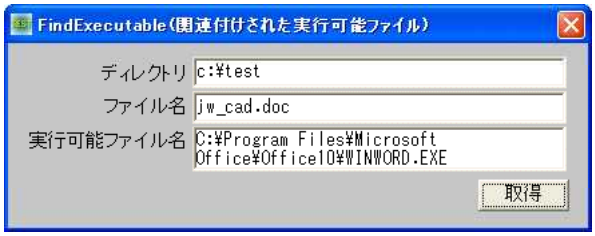
関連付けされた実行可能ファイル取得(Ⅰ)

関連付けされた実行可能ファイルを取得します。
FindExecutable ファイル名に関連付けられている実行可能ファイル名とハンドルを取得

例では、ディレクトリc:¥testにFunction.csv、m2.jtd、jw_cad.docを用意してテストしています。








```
'=======================================
'= 関連付けされた実行可能ファイル取得
'=    (FindExecutable.bas)
'=======================================
#include "Windows.bi"

#define OUT_OF_MEMORY_OR_RESOURCES 0        'メモリまたはリソースが足りない
#define ERROR_FILE_NOT_FOUND 2              'ファイルが見つからない
#define ERROR_PATH_NOT_FOUND 3              'パスが見つからない
```

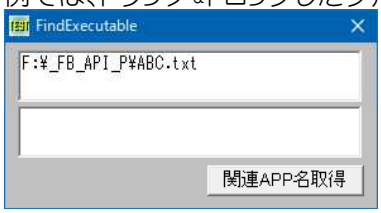
```
#define ERROR_BAD_ForMAT 11                 '不正な形式の実行ファイル
#define NO_ASSOCIATION 31                   '関連付けを取得できない

' ファイル名に関連付けられている実行可能ファイル名とハンドルを取得
Declare Function Api_FindExecutable& Lib "Shell32" Alias "FindExecutableA" (ByVal
lpFile$, ByVal lpDirectory$, ByVal lpResult$)

Var Shared Edit1 As Object
Var Shared Edit2 As Object
Var Shared Text(3) As Object
Var Shared Button1 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Edit2.Attach GetDlgItem("Edit2") : Edit2.SetFontSize 14
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

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

'    Edit1.SetWindowText environ$("windir") & "¥win.ini"
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var FileName As String
    Var DefaultDir As String
    Var ExeFileName As String * 516
    Var ExeInst As Long

    'デフォルトディレクトリを指定
    DefaultDir = GetDlgItemText("Edit1")

    'ファイル名を指定
    FileName = GetDlgItemText("Edit2")

    '関連付けられている実行可能ファイル名を取得
    ExeInst = Api_FindExecutable(FileName, DefaultDir, ExeFileName)

    'インスタンスハンドルにより分岐
    Select Case ExeInst
        Case is > 32

            '実行可能ファイル名を表示
            Text(3).SetWindowText Left$(ExeFileName, InStr(ExeFileName, Chr$(0)) - 1)

        Case OUT_OF_MEMORY_OR_RESOURCES
            Text(3).SetWindowText "メモリまたはリソースが足りません。"

        Case ERROR_FILE_NOT_FOUND
            Text(3).SetWindowText "ファイルが見つかりません。"

        Case ERROR_PATH_NOT_FOUND
            Text(3).SetWindowText "パスが見つかりません。"

        Case ERROR_BAD_ForMAT
            Text(3).SetWindowText "不正な形式の実行ファイルです。"

        Case NO_ASSOCIATION
            Text(3).SetWindowText "関連付けが指定されていません。"
```

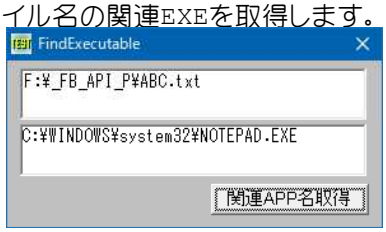
```
        Case Else
            Text(3).SetWindowText "関連付けを取得できません。"
    End Select
End Sub

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

関連付けされた実行可能ファイル取得(II)

関連付けされた実行可能ファイルを取得します。
FindExecutable ファイル名に関連付けられている実行可能ファイル名とハンドルを取得

例では、ドラッグ&ドロップしたファイル名の関連EXEを取得します。





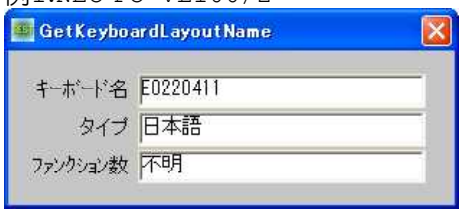

```
'=====================================
'= 関連付けされた実行可能ファイルを取得
'=    (FindExecutable2.bas)
'=====================================
#Include "Windows.bi"

' ファイル名に関連付けられている実行可能ファイル名とハンドルを取得
Declare Function Api_FindExecutable& Lib "shell32" Alias "FindExecutableA" (ByVal
lpFile$, ByVal lpDirectory$, ByVal lpResult$)

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 FileName As String

'=====================================
'=
'=====================================
Declare Function GetFileName(FileName As String) As String
Function GetFileName(FileName As String) As String
    Var Pfad As String
    Var Ret As Long

    Pfad = Space$(256)

    Ret = Api_FindExecutable(FileName, ByVal 0, Pfad)

    If Pfad <> "" Then
        Pfad = Left$(Pfad, InStr(Pfad, Chr$(0)) - 1)
    End If

    If UCase$(Pfad) = UCase$(FileName) Then Pfad = ""

    GetFileName = Pfad
```

```
End Function

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Edit1.SetWindowText FileName
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var sExe As String
    sExe = GetFileName(Edit1.GetWindowText)

    If sExe <> "" Then
        Text1.SetWindowText  sExe
    Else
        Text1.SetWindowText "関連アプリケーションは不明！"
    End If
End Sub

'=====================================
'=
'=====================================
Declare Sub Edit1_SetFocus edecl ()
Sub Edit1_SetFocus()
    Text1.SetWindowText ""
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

    Text1.SetWindowText ""
End Sub

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

キーボード情報を取得

キーボード情報（ キーボード名・タイプ・ファンクションの数）を取得します。
GetKeyboardLayoutName アクティブな入力ロケール識別子を取得
GetKeyboardType キーボードに関する情報を取得

例1:NEC PC-VL100/2



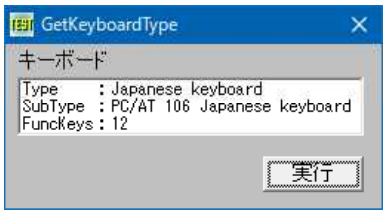
例2:PC-LT23(NOTE)


```
'=====================================
'= キーボード情報を取得
'=    (KeyboardName.bas)
'=====================================
#include "Windows.bi"

#define KL_NAMELENGTH 9
#define KT_Type 0
#define KT_SUBType 1
#define KT_FunctionKEYS 2

' アクティブな入力ロケール識別子を取得
Declare Function Api_GetKeyboardLayoutName& Lib "user32" Alias "GetKeyboardLayoutNameA"
(ByVal pwszKLID$)

' キーボードに関する情報を取得
Declare Function Api_GetKeyboardType& Lib "user32" Alias "GetKeyboardType" (ByVal
nTypeFlag&)
Var Shared Text(5) As Object
For i = 0 To 5
    Text(i).Attach GetDLgItem("Text" & Trim$(Str$(i + 1)))
    Text(i).SetFontSize 14
Next

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var strName As String
    Var kbType As String
    Var fKeys As String
    Var Ret As Long

    strName = String$(KL_NAMELENGTH, 0)

    'キーボード名の取得
    Ret = Api_GetKeyboardLayoutName(strName)

    Select Case Api_GetKeyboardType(KT_Type)
        Case 1
            kbType = "IBM PC/XT 及び互換(83-key)"
        Case 2
            kbType = "Olivetti ?ICO? (102-key)"
        Case 3
            kbType = "IBM PC/AT (84-key) 及び互換"
        Case 4
            kbType = "IBM enhanced (101- Or 102-key)"
        Case 5
            kbType = "Nokia 1050 及び互換"
        Case 6
            kbType = "Nokia 9140 及び互換"
        Case 7
            kbType = "日本語"
        Case Else
            kbType = "不明"
    End Select
    Select Case Api_GetKeyboardType(KT_FunctionKEYS)
        Case 1
            fKeys = "10"
        Case 2
```

```
            fKeys = "12"
        Case 3
            fKeys = "10"
        Case 4
            fKeys = "12"
        Case 5
            fKeys = "10"
        Case 6
            fKeys = "24"
        Case 7
            fKeys = "OEM"
        Case Else
            fKeys = "不明"
    End Select

    Text(3).SetWindowText Left$(strName, InStr(strName, Chr$(0)) - 1)
    Text(4).SetWindowText kbType
    Text(5).SetWindowText fKeys
End Sub

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

キーボードに関する情報取得

GetKeyboardType 現在のキーボードに関する情報を取得




```
'=======================================
'= キーボードに関する情報取得
'=    (GetKeyboardType.bas)
'=======================================
#include "Windows.bi"

' 現在のキーボードに関する情報を取得
Declare Function Api_GetKeyboardType& Lib "user32" Alias "GetKeyboardType" (ByVal
nTypeFlag&)

Var Shared Text1 As Object
Var Shared List1 As Object
Var Shared Button1 As Object

Text1.Attach GetDlgItem("Text1") : Text1.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 KeyboardType As Long
    Var KeyboardSubType As Long
```

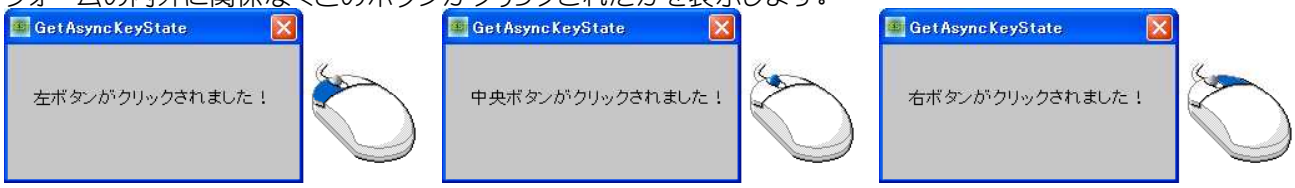
```
Var FunctionKeys As Long

'リストボックスをクリア
List1.Resetcontent

'キーボードタイプを取得
KeyboardType = Api_GetKeyboardType(0)

'キーボードタイプを表示
Select Case KeyboardType
    Case &H1
        List1.AddString "Type    :IBM(R) PC/XT(R) keyboard"
    Case &H2
        List1.AddString "Type    :Olivetti(R) 'ICO' keyboard"
    Case &H3
        List1.AddString "Type    :IBM PC/AT(R) keyboard"
    Case &H4
        List1.AddString "Type    :IBM enhanced keyboard"
    Case &H5
        List1.AddString "Type    :Nokia(R) 1050 keyboard"
    Case &H6
        List1.AddString "Type    :Nokia9140 keyboard"
    Case &H7
        List1.AddString "Type    :Japanese keyboard"
    Case &H8
        List1.AddString "Type    :Korean keyboard"
End Select

'キーボードタイプが日本語文字のキーボードのとき
If KeyboardType = &H7 Then

    'キーボードサブタイプを取得
    KeyboardSubType = Api_GetKeyboardType(1)

    'キーボードサブタイプによって分岐
    Select Case KeyboardSubType

        'Microsoft(DOS/V)
        Case &H0
            List1.AddString "SubType :PC/AT 101 Enhanced keyboard"
        Case &H1
            List1.AddString "SubType :AX compatible keyboard"
        Case &H2
            List1.AddString "SubType :PC/AT 106 Japanese keyboard"
        Case &H3
            List1.AddString "SubType :IBM 5576-002 keyboard"
        Case &H4
            List1.AddString "SubType :IBM 5576-001 keyboard"

        'AX consortium
        Case &H11
            List1.AddString "SubType :AX keyboard"

        'Fujitsu
        Case &H50
            List1.AddString "SubType :FMR JIS Keyboard"
        Case &H51
            List1.AddString "SubType :FMR OASYS Keyboard"
        Case &H52
            List1.AddString "SubType :PC/AT OASYS Keyboard"

        'NEC
        Case &HD01
            List1.AddString "SubType :PC-9800 standard keyboard"
        Case &HD04
            List1.AddString "SubType :PC-9800 note/laptop keyboard"
        Case &HD05
            List1.AddString "SubType :PC-9800 106 Japanese keyboard"
```

```
            'Toshiba
            Case &H1201
                List1.AddString "SubType :Toshiba desktop type keyboard"
            Case &H1202
                List1.AddString "SubType :Toshiba laptop type keyboard"

            '不明
            Case Else
                List1.AddString "SubType :Unknown"
        End Select
    Else
        List1.AddString "SubType :Unjapanese"
    End If

    'ファンクションキー数を取得
    FunctionKeys = Api_GetKeyboardType(2)

    'ファンクションキー数を表示
    List1.AddString "FuncKeys:" & Trim$(Str$(FunctionKeys))
End Sub

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

キーボードの状態を取得

キーボード、マウスボタンの状態を取得します。
GetAsyncKeyState キーボードの状態を取得

フォームの内外に関係なくどのボタンがクリックされたかを表示します。



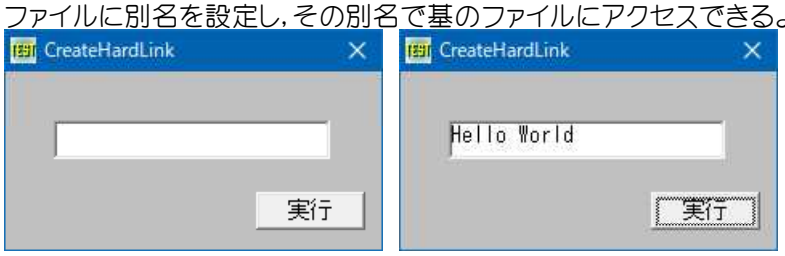

```
'=======================================
'= キーボードの状態を取得
'=    (GetAsyncKeyState.bas)
'=======================================
#include "Windows.bi"

' キーボードの状態を取得
Declare Function Api_GetAsyncKeyState& Lib "user32" Alias "GetAsyncKeyState" (ByVal
vKey&)

#define VK_LBUTTON &H1                    'マウス左ボタン
#define VK_RBUTTON &H2                    'マウス右ボタン
#define VK_MBUTTON &H4                    'マウス中央ボタン

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 50
    Timer1.Enable -1
End Sub

'=====================================
'=
'=====================================
Declare Sub Timer1_Timer edecl ()
Sub Timer1_Timer()
    If Api_GetAsyncKeyState(VK_LBUTTON) Then
        Text1.SetWindowText "左ボタンがクリックされました！"
    Else If Api_GetAsyncKeyState(VK_RBUTTON) Then
        Text1.SetWindowText "右ボタンがクリックされました！"
    Else If Api_GetAsyncKeyState(VK_MBUTTON) Then
        Text1.SetWindowText "中央ボタンがクリックされました！"
    Else
        Text1.SetWindowText ""
    End If
End Sub

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

既存と新規のファイルの間にハードリンクを確立

CreateHardLink 既存ファイルと新規ファイルの間にNTFSハードリンクを確立

ハードリンク
ファイルに別名を設定し，その別名で基のファイルにアクセスできるようにする「リンク」機能の1つ



ファイル1には「Hello」
ファイル2には「World」

```
'=====================================
'= 既存と新規のファイルの間にハードリンクを確立
'=    (CreateHardLink.bas)
'=====================================
#include "Windows.bi"

' 既存ファイルと新規ファイルの間にNTFSハードリンクを確立
Declare Function Api_CreateHardLink& Lib "kernel32" Alias "CreateHardLinkA" (ByVal
lpFileName$, ByVal lpExistingFileName$, ByRef lpSecurityAttributes As Any)

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 sSave As String
```

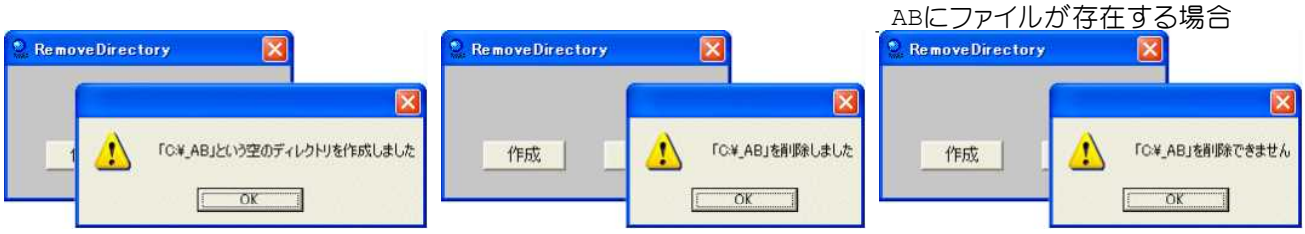
```
    Var Ret As Long

    'Helloと入力したファイルを作成
    Open "test1.txt" For Output As #1
        Print #1, "Hello ";
    Close #1

    'ハードリンク(同じファイルを示す2番目のファイル名)を作成
    Ret = Api_CreateHardLink("test2.txt", "test1.txt", ByVal 0)
    'Worldと入力した2番目のファイルを作成
    Open "test2.txt" For Append As #1
        Print #1, "World"
    Close #1

    'bufferを作成
    sSave = String$(Len("test2.txt"), 0)

    '最初のファイルを読む
    Open "test1.txt" For Input As #1
        Input #1, sSave
    Close #1

    '結果を表示 'Hello World'
    Text1.SetWindowText sSave

    'ファイルを削除
    Kill "test1.txt"
    Kill "test2.txt"
End Sub

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

既存の空のディレクトリを削除

既存の空のディレクトリを削除します。
CreateDirectoryEx ディレクトリの新規作成
CreateDirectory ディレクトリの新規作成
RemoveDirectory 既存の空のディレクトリを削除

左:C:¥に_ABというディレクトリを作成します。　　右:RemoveDirectoryを実行(空なので削除成功)
CreateDirectoryExの場合、第一引数の属性を引き継ぐため、該当するフォルダが存在しない場合は新規フォルダは作成されません。

ABにファイルが存在する場合




```
'=====================================
'= 既存の空のディレクトリを削除
'=    (RemoveDirectory.bas)
'=====================================
#include "Windows.bi"

' 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)
```

```
' ディレクトリの新規作成
'Declare Function Api_CreateDirectory& Lib "kernel32" Alias "CreateDirectoryA" (ByVal
lpPathName$, lpSecurityAttributes As SECURITY_ATTRIBUTES)

' 既存の空のディレクトリを削除
Declare Function Api_RemoveDirectory& Lib "kernel32" Alias "RemoveDirectoryA" (ByVal
lpPathName$)

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

    '新しいディレクトリを作成
    Ret = Api_CreateDirectoryEx("C:¥_FB", "C:¥_AB", ByVal 0)
'   Ret = Api_CreateDirectory("C:¥_AB", ByVal 0)

    If Ret <> 0 Then
        A% = MessageBox("", "「C¥_AB」という空のディレクトリを作成しました", 0, 2)
    Else
        A% = MessageBox("", "「C¥_AB」というディレクトリは作成できません！", 0, 2)
    End If
End Sub

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

    'ディレクトリを削除
    Ret = Api_RemoveDirectory("C:¥_AB")

    If Ret = 1 Then
        A% = MessageBox("", "「C:¥_AB」を削除しました", 0, 2)
    Else
        A% = MessageBox("", "「C:¥_AB」を削除できません", 0, 2)
    End If
End Sub

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

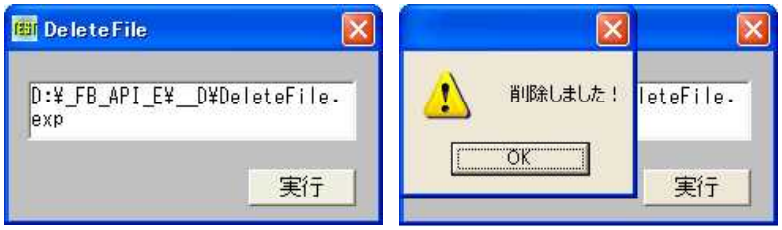
既存のパスにファイル名を追加

PathAppend 既存のパスにファイル名を追加

```
'=====================================
'= 既存のパスにファイル名を追加
'=    (PathAppend.bas)
'=====================================
#include "Windows.bi"

' 既存のパスにファイル名を追加
Declare Function Api_PathAppend& Lib "shlwapi" Alias "PathAppendA" (ByVal pszPath$,
ByVal pMore$)

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:¥testpath¥dir"
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_PathAppend(sSave, "myfile.tst")

    Text1.SetWindowText TrimNull(sSave)
End Sub

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

既存のファイルを削除する

DeleteFile 既存のファイルを削除

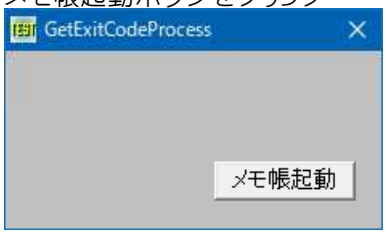



```
'=====================================
'= 既存のファイルを削除する
'=    (DeleteFile.bas)
'=====================================
#include "Windows.bi"

' 既存のファイルを削除
Declare Function Api_DeleteFile& Lib "Kernel32" Alias "DeleteFileA" (ByVal lpFileName$)

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 sFileName As String

'=====================================
'= 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)
    sFileName = GetDropFileName(DF, 0)
    Edit1.SetWindowText sFileName
End Sub

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

    sFileName = Edit1.GetWindowText

    Ret = Api_DeleteFile(sFileName)
    If Ret <> 0 Then
        A% = MessageBox("", "削除しました！", 0, 2)
    Else
        A% = MessageBox("", "削除できません！", 0, 2)
    End If
End Sub

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

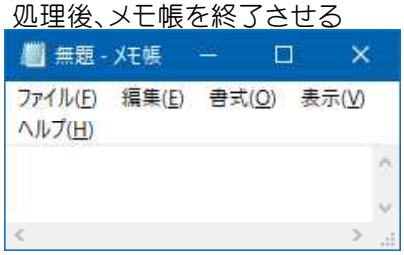
起動させたアプリケーションの終了を知る(Ⅰ)

例ではメモ帳を起動し、それが終了したことをメッセージボックスに表示させています。
WaitForSingleObject 指定された時間が経過するまでスレッドをスリープ
CreateProcess プロセスの起動
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
GetExitCodeProcess 指定プロセスの終了コードを取得

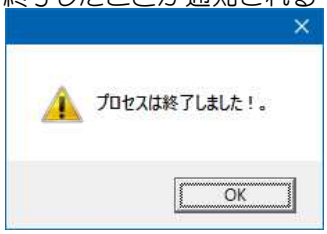
メモ帳起動ボタンをクリック



処理後、メモ帳を終了させる



終了したことが通知される



参考(‥というより丸写し)
http://support.microsoft.com/default.aspx?scid=KB;en-us;q129796

```
'=======================================
'= アプリケーションを起動し終了したことを得る(Ⅰ)
'= 参考:http://support.microsoft.com/default.aspx?scid=KB;en-us;q129796
'=    (GetExitCodeProcess.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

' 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&)

' プロセスの起動
Declare Function Api_CreateProcess& Lib "kernel32" Alias "CreateProcessA" (ByVal aNm$,
ByVal ComLine$, ByVal ProcAttr&, ByVal thAttr&, ByVal Handl&, ByVal Flags&, ByVal
Environ&, ByVal Dir$, StartupInfo As STARTUPINFO, ProcInfo As PROCESS_INFORMATION)

' 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&)

' 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&)

#define NORMAL_PRIORITY_CLASS &H20          '通常クラス(一般的なプロセス)
#define INFINITE &HFFFF                     '無限に中断

Var Shared Button1 As Object

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

'=====================================
'=
'=====================================
Declare Function ExecCmd(cmdline$)
function ExecCmd(cmdline$)
    Var pi As PROCESS_INFORMATION
    Var si As STARTUPINFO
    Var Ret As Long

    'STARTUPINFO構造体の初期化
    si.cb = Len(start)

    'アプリケーション起動
    Ret = Api_CreateProcess(ByVal 0, cmdline$, 0, 0, 1, NORMAL_PRIORITY_CLASS, 0, ByVal 0,
si, pi)

    '起動されたアプリケーションが終了するまで待つ
    Ret = Api_WaitForSingleObject(pi.hProcess, INFINITE)
    Ret = Api_GetExitCodeProcess(pi.hProcess, Ret)
    Ret = Api_CloseHandle(pi.hThread)
    Ret = Api_CloseHandle(pi.hProcess)
    ExecCmd = Ret
End Function

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

    'メモ帳を起動
    Ret = ExecCmd("notepad.exe")
    A% = MessageBox("", "プロセスは終了しました！。終了コード:" & Str$(Ret), 0, 2)
End Sub

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

起動させたアプリケーションの終了を知る(II)

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
GetWindowThreadProcessId ウィンドウのプロセスIDとスレッドIDを取得
PostMessage 指定されたウィンドウを作成した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
GetExitCodeProcess 指定プロセスの終了コードを取得
OpenProcess 既存のプロセスオブジェクトのハンドルを取得

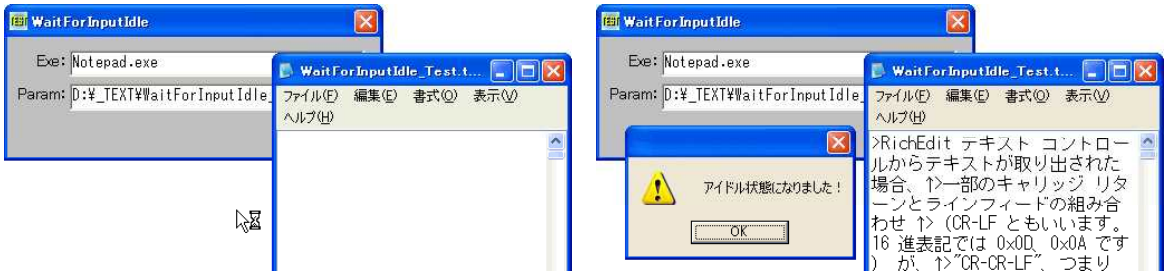

```
'=====================================
'= 起動させたアプリケーションの終了を知る(II)
'=    (KeepWatch.bas)
'=====================================
#include "Windows.bi"

' 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&)

' 指定されたウィンドウを作成した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&, lParam As Any)

' 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&)

' 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&)

#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              '同期をとる
#define STILL_ACTIVE &H103                'プロセス実行中

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 GetID(MyCaption As String) As Long
Function GetID(MyCaption As String) As Long
    Var st As String
    Var ProcessID as long
    Var RetVal As Long
    Var Ret As Long

    RetVal = Api_FindWindow(ByVal 0, "無題 - メモ帳")

    If RetVal Then
        Ret = Api_GetWindowThreadProcessId(RetVal, ProcessId)
        Ret = Api_PostMessage(Ret, WM_CLOSE, 0, 0)

        'プロセスID取得
        GetID = ProcessId
        Exit Function
    End If

    'プロセスID取得できない
    GetID = 0
```

```
End Function

'=====================================
'=
'=====================================
Declare Sub Watch()
Sub Watch()
    Var Flag As Long
    Var CheckID As Long
    Var i As Long
    Var RetVal As Long
    Var Ret As Long

    CheckID = GetID("無題 - メモ帳")

    If CheckID = 0 Then
        Exit Sub
    End if

    Flag = STANDARD_RIGHTS_REQUIRED Or SYNCHRONIZE Or &HFFF

    RetVal = Api_OpenProcess(Flag, False, CheckID)

    'ループして監視
    Do
        Ret = Api_GetExitCodeProcess(RetVal, i)

        If Not (i = STILL_ACTIVE) Then
            Text1.SetWindowtext "フォームはありません！"
            Exit Do
        End If

        CallEvent
        Text1.SetWindowtext "フォームは存在します！"
    Loop
End Sub

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub button1_on()
    Watch
End Sub

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

起動したアプリがアイドル状態になるまで待機

CreateProcess プロセスの起動
WaitForInputIdle タイムアウト時間が経過するまで待機

例では、Exeに「Notepad.exe」を、Paramに意識して重いtextを読ませています。




```
'=======================================
'= 起動したアプリがアイドル状態になるまで待機
'=    (WaitForInputIdle.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 Name$,
ByVal Cmd$, pAtt As Any, tAtt As Any, ByVal Hand&, ByVal Flg&, Env As Any, ByVal Dir$, sInfo
As Any, pInfo As PROCESS_INFORMATION)

' タイムアウト時間が経過するまで待機
Declare Function Api_WaitForInputIdle& Lib "user32" Alias "WaitForInputIdle" (ByVal
hProcess&, ByVal dwMilliseconds&)

#define NORMAL_PRIORITY_CLASS &H20          '通常クラス(一般的なプロセス)
#define STARTF_USESHOWWINDOW &H1
#define SW_SHOW 5                           'ウィンドウをアクティブ化し現在の位置とサイズで表示

Var Shared Edit(1) As Object
Var Shared Text(1) As Object
Var Shared Button1 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
Next
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

Var Shared sPath As String
Var Shared sParam As String
```

```
'====================================
'=
'====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var si As STARTUPINFO
    Var pi As PROCESS_INFORMATION
    Var Ret As Integer

    sPath = Edit(0).GetWindowText
    sParam = Edit(1).GetWindowText

    On Error GoTo *Err_Trap

    ShellEx = False

    si.cb = Len(si)
    si.dwFlags = STARTF_USESHOWWINDOW
    si.wShowWindow = SW_SHOW

    'アプリケーションを起動しプロセス作成する
    If Api_CreateProcess(ByVal 0, sPath & " " & sParam, ByVal 0, ByVal 0, 1,
NORMAL_PRIORITY_CLASS, ByVal 0, ByVal 0, si, pi) = 0 Then GoTo *Err_Trap

    Do
        If Api_WaitForInputIdle(pi.hProcess, 100) = 0 Then Exit Do
    Loop

    ShellEx = True

    A% = MessageBox("", "アイドル状態になりました！", 0, 2)
    Exit Sub
    *Err_Trap
        A% = MessageBox("", "エラー", 0, 2)
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)
    Edit(0).SetWindowText sPath
End Sub

'====================================
'= シェルドロップされたパラメータを取得
'====================================
Declare Sub Edit2_DropFiles edecl (ByVal DF As Long)
Sub Edit2_DropFiles(ByVal DF As Long)
    Var Ret As Long

    Ret = GetDropFileCount(DF)
    sParam = GetDropFileName(DF, 0)
    Edit(1).SetWindowText sParam
End Sub

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

起動モードの取得

起動モードを取得します。
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
SM_CLEANBOOT(67) Windowsを起動した方法

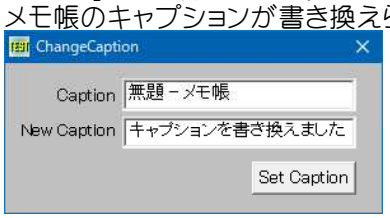
Windows2000機でテストしてみました。


```
'=======================================
'= 起動モードを調べる
'=    (SM_CLEANBOOT.bas)
'=======================================

' 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 SM_CLEANBOOT 67                         'Windowsを起動した方法
Select Case Api_GetSystemMetrics(SM_CLEANBOOT)

    Case 1
        Print "セーフモードで起動"
    Case 2
        Print "ネットワークを介したセーフモードで起動"
    Case Else
        Print "通常モードで起動"
End Select

Stop
End
```

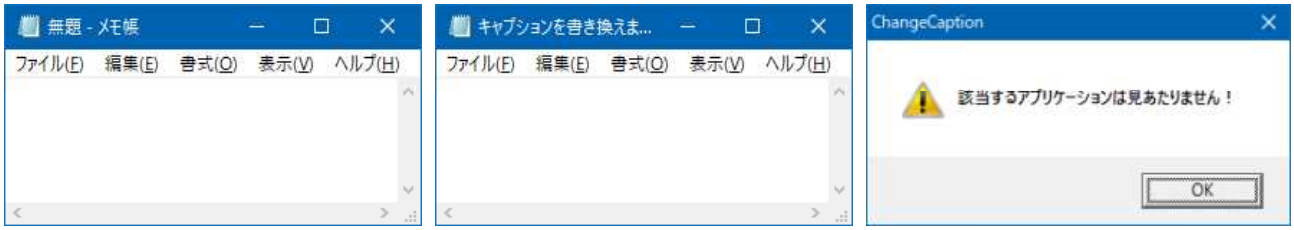
キャプション(アプリケーションタイトル)を変更

アプリケーションのタイトルを変更します。
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
SendMessage ウィンドウにメッセージを送信
WM_SETTEXT(&HC) ウィンドウ(コントロール)のタイトル・テキストを変更

例では、メモ帳のタイトルを変更します。メモ帳の起動時キャプションは、無題 - メモ帳になっています。
New Caption欄に新しいタイトル(例:キャプションを書き換える)を入力し、『Set Caption』ボタンをクリックすると、メモ帳のキャプションが書き換えられます。



New Caption欄にキャプションを書き換えると入力し、『Set Caption』ボタンをクリック
キャプションが書き換えられた状態。該当する文字列がない場合はその旨表示されます。




```
'=====================================
'= キャプション(アプリケーションタイトル)を変更
'=    (ChangeCaption.bas)
'=====================================
#include "Windows.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のハンドルを返す
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&, lParam As Any)

#define WM_SETTEXT &HC                          'ウインドウ(コントロール)のタイトル・テキストを変更
#define vbNullString ByVal 0

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var target_hwnd As Long
    Var target_name As String
    Var new_caption As String
    Var Ret As Long
    target_name = Edit1.GetWindowText
    target_hwnd = Api_FindWindow(vbNullString, target_name)

    If target_hwnd = 0 Then
        A% = MessageBox(GetWindowText, "該当するアプリケーションは見あたりません！", 0 ,2)
        Exit Sub
    End If

    new_caption = GetDlgItemText("Edit2")
    Ret = Api_SendMessage(target_hwnd, WM_SETTEXT, 0, new_caption)
End Sub

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

キャレットの現在位置を取得

キャレットの現在位置を取得表示します。
GetCaretPos キャレットの現在位置を取得

キャレット位置をタイトルバーに表示しています。

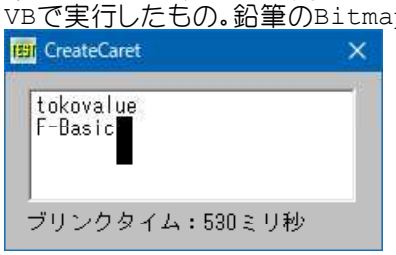
キャレットは赤で表示しています。

```
'=====================================
'= キャレットの現在位置を取得
'=    (GetCaretPos.bas)
'=====================================
#include "Windows.bi"

Type POINTAPI
    X   As Long
    Y   As Long
End Type

' キャレットの現在位置を取得
Declare Function Api_GetCaretPos& Lib "user32" Alias "GetCaretPos" (lpPoint As POINTAPI)

Var Shared Edit1 As Object
Edit1.Attach GetDLgItem("Edit1") : Edit1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Function GetEdCursX() As Long
Function GetEdCursX() As Long
    Var pt As POINTAPI
    Var Ret As Long

    Ret = Api_GetCaretPos(pt)
    GetEdCursX = pt.X
End Function

'=====================================
'=
'=====================================
Declare Function GetEdCursY() As Long
Function GetEdCursY() As Long
    Var pt As POINTAPI
    Var Ret As Long

    Ret = Api_GetCaretPos(pt)
    GetEdCursY = pt.Y
End Function

'=====================================
'=
'=====================================
Declare Sub Edit1_Change edecl ()
Sub Edit1_Change()
    Var XPos As Long
    Var YPos As Long

    XPos = GetEdCursX
    YPos = GetEdCursY
    SetWindowText "X:" & Trim$(Str$(XPos)) & "/Y:" & Trim$(Str$(YPos))
End Sub

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

キャレットを作成する

独自のキャレットを作成します。
CreateCaret独自のキャレットを作成する
ShowCaret キャレットを表示する
GetCaretBlinkTime キャレットのブリンク時間を取得する

キャロットサイズを幅10、高さ30に設定しています。
VBで実行したもの。鉛筆のBitmapを作成(色を反転しています)しキャレットとしています



※VBでのコード

```
    Sub Text1_GotFocus()
        hWnd& = GetFocus&()
        hBitmap& = Picture1.Picture
        Call CreateCaret(hWnd&, hBitmap&, 0, 0)
        X& = ShowCaret&(h&)
    End Sub
```

参照
    アイコンをカーソルに･カーソル(キャレット)をBitmapで

```
'========================================
'= キャレットを作成する
'=    (Createcaret.bas)
'========================================
#include "Windows.bi"

' 独自のキャレットを作成する
Declare Function Api_CreateCaret& Lib "user32" Alias "CreateCaret" (ByVal hWnd&, ByVal
hBitmap&, ByVal nWidth&, ByVal nHeight&)

' キャレットを表示する
Declare Function Api_ShowCaret& Lib "user32" Alias "ShowCaret" (ByVal hWnd&)

' キャレットのブリンク時間を取得する
Declare Function Api_GetCaretBlinkTime& Lib "user32" Alias "GetCaretBlinkTime" ()
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 Sub MainForm_Start edecl ()
Sub MainFOrm_Start()
    Text1.SetWindowText "ブリンクタイム:" & Trim$(Str$(Api_GetCaretBlinkTime)) & "ミリ秒"
    Edit1.SetFocus
End Sub

'========================================
'=
'========================================
Declare Sub Edit1_SetFocus edecl ()
Sub Edit1_SetFocus ()
    Var hWnd As Long
    Var hImage As Long
    Var nWidth As Long
    Var nHeight As Long
    Var Ret As Long
```

```
    hWnd = Edit1.GethWnd
    nWidth = 10
    nHeight = 30

    Ret = Api_CreateCaret(hWnd, 0, nWidth, nHeight)
    Ret = Api_ShowCaret(hWnd)
End Sub

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

キャレットの表示・非表示

キャレット(カーソル)を表示・非表示します。
ShowCaret キャレットを表示する
HideCaret キャレットを非表示

キャレットは赤で表示しています。

```
'=====================================
'= キャレットの表示・非表示
'=    (HideCaret.bas)
'=====================================
#include "Windows.bi"

' キャレットを表示する
Declare Function Api_ShowCaret& Lib "user32" Alias "ShowCaret" (ByVal hWnd&)

' キャレットを非表示にする
Declare Function Api_HideCaret& Lib "user32" Alias "HideCaret" (ByVal hWnd&)

Var Shared Edit(1) As Object
Var Shared Text(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
Next

'=====================================
'=
'=====================================
Declare Sub Edit2_SetFocus edecl ()
Sub Edit2_SetFocus()
    Var Ret As Long

    'キャレットを非表示
    Ret = Api_HideCaret(Edit(1).GethWnd)
End Sub

'=====================================
'=
'=====================================
Declare Sub Edit2_KillFocus edecl ()
```

```
Sub Edit2_KillFocus()
    Var Ret As Long

    'キャレットを表示
    Ret = Api_ShowCaret(Edit(1).GethWnd)
End Sub

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

キャレットのブリンクタイム設定

SetCaretBlinkTime ブリンクタイム設定
GetCaretBlinkTime ブリンクタイム取得

ブリンクタイムを入力(単位ミリ秒)し、『変更』ボタンをクリックします。
カーソルの点滅速度を確認してください。『×』クリックで元の点滅時間に戻しています。

初期状態(図参照)では、800という値でした

```
'=======================================
'= ブリンクタイムの設定
'=    (SetCaretBlinkTime.bas)
'=======================================
#include "Windows.bi"

' キャレットの点滅時間を設定
Declare Function Api_SetCaretBlinkTime& Lib "user32" Alias "SetCaretBlinkTime" (ByVal
wMSeconds&)

' キャレットのブリンク時間を取得する
Declare Function Api_GetCaretBlinkTime& Lib "user32" Alias "GetCaretBlinkTime" ()

Var Shared Edit1 As Object
Var Shared OldBT As Long

'=======================================
'=
'=======================================
Declare Sub Mainform_Start edecl ()
Sub Mainform_Start()
    OldBT = Api_GetCaretBlinkTime          '元のブリンクタイムを取得保持
```

```
    Edit1.Attach GetDlgItem("Edit1")
    Edit1.SetWindowText Trim$(Str$(OldBT))
    Edit1.SetFocus
End Sub

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

    NewBT = val(GetDlgItemText("Edit1"))    '単位ミリ秒
    Ret = Api_SetCaretBlinkTime(NewBT)      'Newブリンクタイム設定
    Edit1.SetFocus
End Sub

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

    If Cancel% = 0 Then
        Ret = Api_SetCaretBlinkTime(OldBT) '元のブリンクタイムに戻す
        End
    End If
End Sub

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

キャレット幅の取得と設定

SystemParametersInfo システム全体に関するパラメータを取得・設定
SPI_GETCARETWIDTH(&H2006) キャレット幅を取得
SPI_SETCARETWIDTH(&H2007) キャレット幅を設定
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

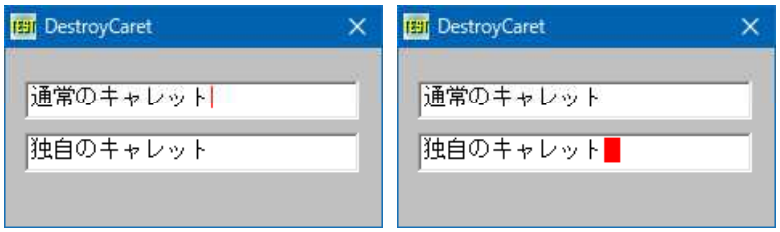

```
'=====================================
'= キャレット幅の取得と設定
'=    (CaretWidth.bas)
'=====================================
#include "Windows.bi"

' 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_GETCARETWIDTH &H2006            'キャレット幅を取得
#define SPI_SETCARETWIDTH &H2007            'キャレット幅を設定
#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 Edit1 As Object
Var Shared Text1 As Object
Var Shared Button1 As Object
Var Shared Button2 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14

Var Shared OldCaretWidth As Long

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

    Edit1.SetWindowText "F-Basic Programming Tips"

    Ret = Api_SystemParametersInfo(SPI_GETCARETWIDTH, 0, OldCaretWidth, 0)
End Sub

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

    'キャレットの幅を指定
    CaretWidth = 10

    'キャレットの幅を表示
    Text1.SetWindowtext "Caret Width =" & Str$(CaretWidth) & " Pixel"

    'キャレットの幅を設定
    Ret = Api_SystemParametersInfo(SPI_SETCARETWIDTH, 0, ByVal CaretWidth,
SPIF_UPDATEINIFILE Or SPIF_SENDWININICHANGE)

    Edit1.SetFocus
End Sub

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

    'キャレットの幅を表示
    Text1.SetWindowtext "Caret Width =" & Str$(OldCaretWidth) & " Pixel"

    'キャレットの幅を設定
    Ret = Api_SystemParametersInfo(SPI_SETCARETWIDTH, 0, ByVal OldCaretWidth,
SPIF_UPDATEINIFILE Or SPIF_SENDWININICHANGE)

    Edit1.SetFocus
End Sub

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

```
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
    Button2_on
End Sub

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

キャレットを破棄する

CreateCaret 独自のキャレットを作成する
DestroyCaret キャレットを破棄する
HideCaret キャレットを非表示にする
ShowCaret キャレットを表示する



キャレットを赤で表示しています。

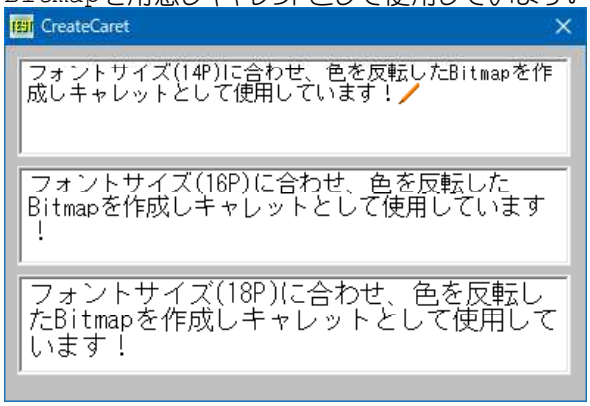
```
'=====================================
'= キャレットを破棄する
'=    (DestroyCaret.bas)
'=====================================
#include "Windows.bi"

' 独自のキャレットを作成する
Declare Function Api_CreateCaret& Lib "user32" Alias "CreateCaret" (ByVal hWnd&, ByVal
hBitmap&, ByVal nWidth&, ByVal nHeight&)

' キャレットを破棄する
Declare Function Api_DestroyCaret& Lib "user32" Alias "DestroyCaret" ()

' キャレットを非表示にする
Declare Function Api_HideCaret& Lib "user32" Alias "HideCaret" (ByVal hWnd&)

' キャレットを表示する
Declare Function Api_ShowCaret& Lib "user32" Alias "ShowCaret" (ByVal hWnd&)

Var Shared Edit1 As Object
Var Shared Edit2 As Object

Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Edit2.Attach GetDlgItem("Edit2") : Edit2.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Edit2_SetFocus edecl ()
Sub Edit2_SetFocus()
    Var hCaret As Long
    Var CaretWidth As Long
    Var CaretHeight As Long
    Var Ret As Long

    'キャレット情報を指定
    hCaret = 0
```

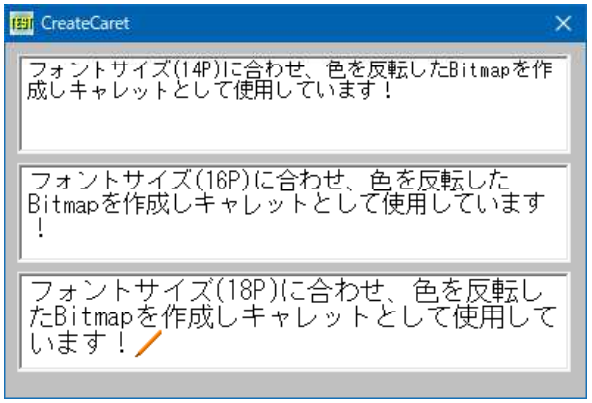
```
    CaretWidth = 10
    CaretHeight = 15

    '独自のキャレットを作成
    Ret = Api_CreateCaret(Edit2.GethWnd, hCaret, CaretWidth, CaretHeight)

    '作成したキャレットを表示
    Ret = Api_ShowCaret(Edit2.GethWnd)
End Sub

'=======================================
'=
'=======================================
Declare Sub Edit2_KillFocus edecl ()
Sub Edit2_KillFocus()
    Var Ret As Long

    'キャレットを非表示
    Ret = Api_HideCaret(EDit2.GethWnd)
End Sub

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

    '作成したキャレットを破棄
    Ret = Api_DestroyCaret()
End Sub

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

キャレットをBitmapで

ビットマップファイルをキャレットとして使用します。
CreateCaret 独自のキャレットを作成
GetCaretBlinkTime キャレットのブリンク時間を取得
SetCaretBlinkTime キャレットのブリンクタイムを設定
ShowCaret キャレットを表示する
LoadImage 画像ファイルを読み込む

例では、フォントサイズを14P、16P、18PとしたEditBoxにそれぞれのフォントサイズに合わせた鉛筆(のつもり••)のBitmapを用意しキャレットとして使用しています。





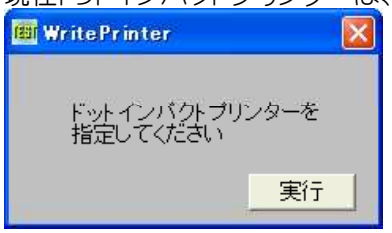
```
'====================================
'= カーソル(キャレット)をBitmapで
'=    (CreateCaret3.bas)
'====================================
#include "Windows.bi"

' 独自のキャレットを作成する
Declare Function Api_CreateCaret& Lib "user32" Alias "CreateCaret" (ByVal hWnd&, ByVal
hBitmap&, ByVal nWidth&, ByVal nHeight&)

' キャレットのブリンク時間を取得する
Declare Function Api_GetCaretBlinkTime& Lib "user32" Alias "GetCaretBlinkTime" ()

' キャレットの点滅時間を設定
Declare Function Api_SetCaretBlinkTime& Lib "user32" Alias "SetCaretBlinkTime" (ByVal
wMSeconds&)
' キャレットを表示する
Declare Function Api_ShowCaret& Lib "user32" Alias "ShowCaret" (ByVal hWnd&)

' 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 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

Var Shared OldBlinkTime As Long
Var Shared hBitmap As Long
Var Shared Ret As Long

Var Shared Edit1 As Object
Var Shared Edit2 As Object
Var Shared Edit3 As Object
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14
Edit2.Attach GetDlgItem("Edit2") : Edit2.SetFontSize 16
Edit3.Attach GetDlgItem("Edit3") : Edit3.SetFontSize 18

'====================================
'=
'====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    OldBlinkTime = Api_GetCaretBlinkTime
    Ret = Api_SetCaretBlinkTime(400)

    hBitmap = Api_LoadImage(GethInst, "Pen14.bmp", IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE)
    txt$ = "フォントサイズ(14*14)に合わせ、色を反転したBitmapを作成しキャレットとして使用しています！"
    Edit1.SetWindowText txt$
    Edit1.SetSelText Len(txt$), Len(txt$)
    Edit1.SetFocus
End Sub

'====================================
'=
'====================================
Declare Sub Edit1_SetFocus edecl ()
Sub Edit1_SetFocus()
    hBitmap = Api_LoadImage(GethInst, "Pen14.bmp", IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE)
    txt$ = "フォントサイズ(14P)に合わせ、色を反転したBitmapを作成しキャレットとして使用しています！"
    Edit1.SetWindowText txt$
    Edit1.SetSelText Len(txt$), Len(txt$)
    Edit1.SetFocus
    Ret = Api_CreateCaret(Edit1.GethWnd, hBitmap, 0, 0)
    Ret = Api_ShowCaret(Edit1.GethWnd)
End Sub

'====================================
'=
'====================================
Declare Sub Edit2_SetFocus edecl ()
```

```
Sub Edit2_SetFocus()
    hBitmap = Api_LoadImage(GethInst, "Pen16.bmp", IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE)
    txt$ = "フォントサイズ(16P)に合わせ、色を反転したBitmapを作成しキャレットとして使用しています！"
    Edit2.SetWindowText txt$
    Edit2.SetSelText Len(txt$), Len(txt$)
    Edit2.SetFocus
    Ret = Api_CreateCaret(Edit2.GethWnd, hBitmap, 0, 0)
    Ret = Api_ShowCaret(Edit2.GethWnd)
End Sub

'=====================================
'=
'=====================================
Declare Sub Edit3_SetFocus edecl ()
Sub Edit3_SetFocus()
    hBitmap = Api_LoadImage(GethInst, "Pen18.bmp", IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE)
    txt$ = "フォントサイズ(18P)に合わせ、色を反転したBitmapを作成しキャレットとして使用しています！"
    Edit3.SetWindowText txt$
    Edit3.SetSelText Len(txt$), Len(txt$)
    Edit3.SetFocus
    Ret = Api_CreateCaret(Edit3.GethWnd, hBitmap, 0, 0)
    Ret = Api_ShowCaret(Edit3.GethWnd)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
   Ret = Api_SetCaretBlinkTime(OldBlinkTime)
End Sub

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

行単位で印刷する方法

ドットインパクトプリンターにのみ使用できます。
OpenPrinter プリンタオブジェクトをオープン
ClosePrinter プリンタオブジェクトを閉じる
EndDocPrinter 印刷ジョブを終了
EndPagePrinter 指定されたプリンタのページの終端と次のページの先頭を示す
StartDocPrinter 印刷ジョブを開始
StartPagePrinter プリンタドライバがデータを受け取る準備をさせる
WritePrinter 指定されたプリンタにデータを書き込むよう印刷スプーラに通知

現在ドットインパクトプリンターは、持ちあわせがないので試験できません(^^;)




```
'=====================================
'= 行単位で印刷する方法(ドットインパクトプリンターのみ)
'=    (WritePrinter.bas)
'=    http://support.microsoft.com/?kbid=175083
'=====================================
```

```
#include "Windows.bi"

Type DOCINFO
    pDocName          As Long
    pOutputFile       As Long
    pDatatype         As Long
End Type

' プリンタオブジェクトをオープン
Declare Function Api_OpenPrinter& Lib "winspool.drv" Alias "OpenPrinterA" (ByVal
pPrinterName$, phPrinter&, ByVal pDefault&)

' プリンタオブジェクトを閉じる
Declare Function Api_ClosePrinter& Lib "winspool.drv" Alias "ClosePrinter" (ByVal
hPrinter&)

' 印刷ジョブを終了
Declare Function Api_EndDocPrinter& Lib "winspool.drv" Alias "EndDocPrinter" (ByVal
hPrinter&)

' 指定されたプリンタのページの終端と次のページの先頭を示す
Declare Function Api_EndPagePrinter& Lib "winspool.drv" Alias "EndPagePrinter" (ByVal
hPrinter&)

' 印刷ジョブを開始
Declare Function Api_StartDocPrinter& Lib "winspool.drv" Alias "StartDocPrinterA" (ByVal
hPrinter&, ByVal Level&, pDocInfo 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_StartPagePrinter& Lib "winspool.drv" Alias "StartPagePrinter"
(ByVal hPrinter&)

' 指定されたプリンタにデータを書き込むよう印刷スプーラに通知
Declare Function Api_WritePrinter& Lib "winspool.drv" Alias "WritePrinter" (ByVal
hPrinter&, pBuf As Any, ByVal cdBuf&, pcWritten&)

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 hPrinter As Long

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var Doc As Long
    Var di As DOCINFO
    Var Ret As Long

    Ret = Api_OpenPrinter("PC-PR201", hPrinter, 0)

    If Ret = 0 Then
        A% = MessageBox("", "プリンタを認識できません！", 0, 2)
        Exit Sub
    End If

    di.pDocName = StrAdr("Test Print" & Chr$(0))
    di.pOutputFile = StrAdr(Chr$(0))
    di.pDatatype = StrAdr(Chr$(0))

    Doc = Api_StartDocPrinter(hPrinter, 1, di)
    Ret = Api_StartPagePrinter(hPrinter)
End Sub
```

```
'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var lpcWritten As Long
    Var txt As String
    Var Ret As Long

    txt = "How's that for Magic !!!!" & Chr$(13, 10)
    Ret = Api_WritePrinter(hPrinter, txt, Len(txt), lpcWritten)
End Sub

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

    Ret = Api_EndPagePrinter(hPrinter)
    Ret = Api_EndDocPrinter(hPrinter)
    Ret = Api_ClosePrinter(hPrinter)
End Sub

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

協定世界時間とローカル時間との分差

協定世界時間とローカル時間との分差を求めます。
GetTimeZoneInformation タイムゾーン情報を取得


```
'=====================================
'= 協定世界時間とローカル時間との分差
'=    (GetTimeZoneInformation.bas)
'=====================================

' 日付と時刻を定義する構造体
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

' タイムゾーンを定義する構造体
Type TIME_ZONE_INFORMATION
    Bias              As Long               '協定世界時(UTC)と現地時間の分差
```

```
    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)

Var tzi As TIME_ZONE_INFORMATION
Var Ret  As Long

' タイムゾーン情報を取得
Ret = Api_GetTimeZoneInformation(tzi)

' 協定世界時間とローカル時間との分差を表示
Print "協定世界時間とローカル時間との分差:" & Str$(tzi.Bias) & "分"

Stop
End
```

綺麗に縮小転送

SetStretchBltModeを使って画像を綺麗に縮小転送します。
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
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

例では、Picture1の画像をPicture2に StretchBlt転送、Picture3に SetStretchBltModeで
COLORONCOLOR を設定後StretchBlt 転送しています。
画像比較（違いが判別できるでしょうか。StretchBltのみでは黒が潰れているように見えます。）

それぞれの拡大比較

```
'=====================================
'= 画像を綺麗に縮小
'=    (StretchBltMode.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&)

' 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
#define COLORONCOLOR 3

Var Shared Text1 As Object : Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Var Shared Text2 As Object : Text2.Attach GetDlgItem("Text2") : Text2.SetFontSize 14
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 Bitmap As Object
BitmapObject Bitmap

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    Var hDC1 As Long
    Var hDC2 As Long
    Var hDC3 As Long
    Var Ret As Long

    hDC1 = Api_GetDC(Picture1.GethWnd)
    hDC2 = Api_GetDC(Picture2.GethWnd)
    hDC3 = Api_GetDC(Picture3.GethWnd)

    Bitmap.LoadFile "flower.bmp"
    Picture1.DrawBitmap Bitmap, 0, 0
    Bitmap.DeleteObject

    Ret = Api_StretchBlt(hDC2, 0, 0, Picture2.GetWidth, Picture2.GetHeight, hDC1, 0, 0,
Picture1.GetWidth, Picture1.GetHeight, SRCCOPY)

    Ret = Api_SetStretchBltMode(hDC3, COLORONCOLOR)
    Ret = Api_StretchBlt(hDC3, 0, 0, Picture3.GetWidth, Picture3.GetHeight, hDC1, 0, 0,
Picture1.GetWidth, Picture1.GetHeight, SRCCOPY)
    Ret = Api_ReleaseDC(Picture1.GethWnd, hDC1)
    Ret = Api_ReleaseDC(Picture2.GethWnd, hDC2)
    Ret = Api_ReleaseDC(Picture3.GethWnd, hDC3)
End Sub

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

矩形に3D効果を与える(Ⅰ)

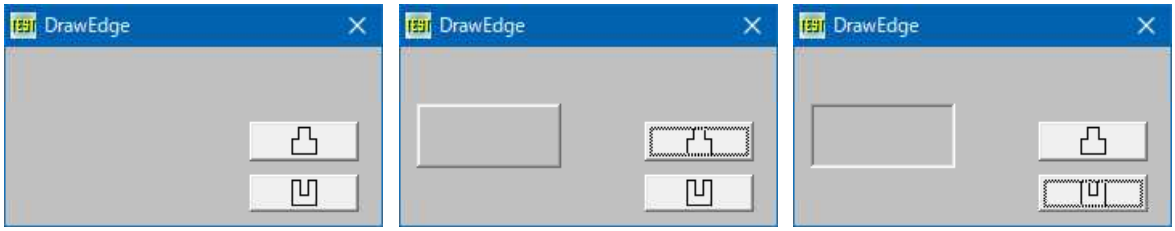
3D効果矩形ラインを引きます。文字出力はDrawTextを用いています。
下の矩形ラインは、F-Basicのコマンドで描画しています。
DrawEdge　矩形に3D効果を与える
DrawText 文字列を指定領域に出力
OffsetRect 矩形領域の補正

```
'====================================
'= 矩形に3D効果を与える(Ⅰ)
'=    (DrawEdge.bas)
'====================================
#include "Windows.bi"

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

' 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&)

' 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&)

' 矩形領域の補正
Declare Function Api_OffsetRect& Lib "user32" Alias "OffsetRect" (lpRect As RECT, ByVal
x&, ByVal y&)

' 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&)

#define DC_ACTIVE 1                          'アクティブなときの色
#define DC_NOTACTIVE &H2                     '
#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 BDR_SUNKENOUTER &H2                  '
#define BDR_RAISEDINNER &H4                  '
#define BDR_RAISED &H5                       '
#define EDGE_ETCHED &H2 Or &H4               '(BDR_SUNKENOUTER Or BDR_RAISEDINNER)
#define BF_LEFT &H1                          '
#define BF_TOP &H2                           '
#define BF_RIGHT &H4                         '
#define BF_BOTTOM &H8                        '
#define BF_RECT &h1 Or &H2 Or &H4 Or &H8     '(BF_LEFT Or BF_TOP Or BF_RIGHT
                                              Or BF_BOTTOM)
#define DFC_BUTTON 4                         'ボタン
#define DFC_POPUPMENU 5                      'ポップアップメニュー
#define DFCS_BUTTON3STATE 8                  '3ステートボタン
#define DT_CENTER &H1                        'テキストを水平方向の中央に揃える

#define TRANSPARENT 1                        '背景色を設定しない

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

    hDC = Api_GetDC(GethWnd)

    rct.Left = 30
    rct.Right = 200
```

```
    rct.Top = 10
    rct.Bottom = 50

    Ret = Api_DrawEdge(hDC, RCT, EDGE_ETCHED, BF_RECT)
    Ret = Api_OffsetRect(RCT, 0, 10)
    txt$ = "DrawEdgeで描画"
    Ret = Api_SetBkMode(hDC, TRANSPARENT)
    Ret = Api_DrawText(hDC, txt$, Len(txt$), rct, DT_CENTER)

    Line(30, 55) - (199, 94), , 1, b
    Line(31, 56) - (200, 95), , 15, b
    Symbol(75, 65), "Lineで描画", 1, 1

    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

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

矩形に3D効果を与える(II)

DrawEdge 矩形に3D効果を与える
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




```
'=======================================
'= 矩形に3D効果を与える(II)
'=    (DrawEdge2.bas)
'=======================================
#include "Windows.bi"

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

' 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&)

' 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 BF_LEFT &H1                         '左辺を描画
#define BF_TOP &H2                          '上辺を描画
#define BF_RIGHT &H4                        '右辺に描画
#define BF_BOTTOM &H8                       '矩形の下辺を描画
```

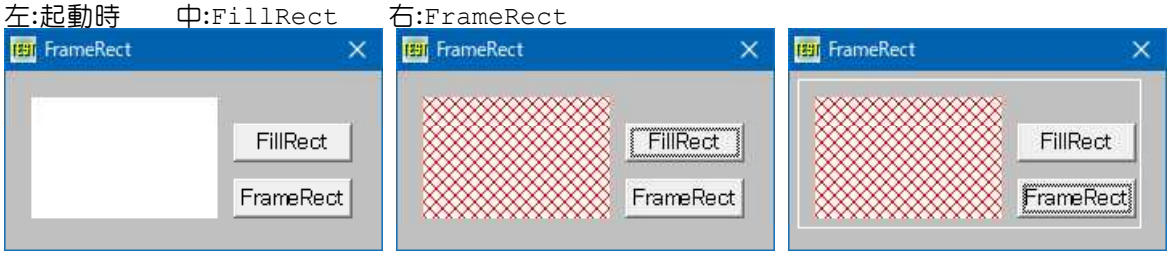
```
#define BF_RECT &HF                         '上下左右の辺を描画

Var Shared hDC As Long

'=====================================
'=
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    hDC = Api_GetDC(GethWnd)
End Sub

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

    rc.Left = 10
    rc.Top = 35
    rc.Right = 100
    rc.Bottom = 74

    Ret = Api_DrawEdge(hDC, rc, BF_LEFT Or BF_RIGHT, BF_RECT)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var rc As RECT
    Var Ret As Long
    rc.Left = 10
    rc.Top = 35
    rc.Right = 100
    rc.Bottom = 74
    Ret = Api_DrawEdge(hDC, rc, BF_TOP Or BF_BOTTOM, BF_RECT)
End Sub

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

    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

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

矩形領域の境界線を描画

長方形の境界線を描画します。
GetSystemMetrics システムの現在の構成を取得
SetRect RECT構造体の値を設定
CreateHatchBrush ハッチパターンの論理ブラシを作成

FillRect ブラシで矩形領域を塗りつぶす
FrameRect 矩形の周囲に指定のブラシで境界線を描画
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を解放
DeleteObject システムリソースを解放

例では、作成したブラシで領域内を塗り潰し、さらにフォームの5ドット内側の矩形領域に境界線を描画します。
左:起動時　　中:FillRect　　右:FrameRect



VBでのFrameRectは下図のようにDIAGCROSSで描画されるのですが、FBでは色の変化はありません。


```
'=====================================
'= 矩形領域の境界線を描画
'=    (FrameRect.bas)
'=====================================
#include "Windows.bi"

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

' さまざまなシステムメトリックの値(表示要素の幅と高さ)とシステムの現在の構成を取得。表示要素とは、ウィンドウの一部、またはシステムが表示する画面の一部を意味する。すべてのサイズをピクセル単位で取得
Declare Function Api_GetSystemMetrics& Lib "user32" Alias "GetSystemMetrics" (ByVal nIndex&)

' 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&)

' ハッチパターンの論理ブラシを作成
Declare Function Api_CreateHatchBrush& Lib "gdi32" Alias "CreateHatchBrush" (ByVal nIndex&, ByVal crColor&)

' 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&)

' 矩形の周囲に指定のブラシで境界線を描画
Declare Function Api_FrameRect& Lib "user32" Alias "FrameRect" (ByVal hDC&, ByRef r As RECT, hBrush&)

' 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&)

' 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&)
```

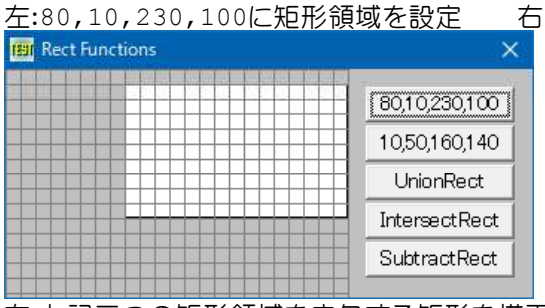
```
#define HS_BDIAGONAL 3                          '斜線(左上-右下)
#define HS_CROSS 4                              '水平と垂直クロスハッチ
#define HS_DIAGCROSS 5                          '45度のクロスハッチ
#define HS_FDIAGONAL 2                          '45度下向きのハッチ(左から右へ)
#define HS_HORIZONTAL 0                         '水平ハッチ
#define HS_VERTICAL 1                           '垂直ハッチ

#define vbRed &H0000FF                          '赤のカラーコード
#define vbBlue &HFF0000                         '青のカラーコード
#define vbGreen &H00FF00                        '緑のカラーコード

#define 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

Var Shared Picture1 As Object
Var SHared Button1 As Object
Var Shared Button2 As Object

Picture1.Attach GetDlgItem("Picture1")
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14
Var Shared rct As RECT
Var Shared hpDC As Long
Var Shared hfDC As Long

'=====================================
'= Picture1およびMainFormのDC取得
'=====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    hfDC = Api_GetDC(GethWnd)
    hpDC = Api_GetDC(Picture1.GethWnd)
End Sub

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

    rct.Left = 0
    rct.Top = 0
    rct.Right = Picture1.GetWidth
    rct.Bottom = Picture1.GetHeight

    hRedBrush = Api_CreateHatchBrush(HS_DIAGCROSS, vbRed)
    Ret = Api_FillRect(hpDC, rct, hRedBrush)

    Ret = Api_DeleteObject(hRedBrush)
End Sub

'=====================================
'= FrameRect(5ドットフォームの内側)
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var fWidth As Long
    Var fHeight As Long
    Var RndCol As Long
    Var hBrush As Long
    Var Ret As Long

    fWidth = Api_GetSystemMetrics(SM_CXFRAME) * 2
    fHeight = Api_GetSystemMetrics(SM_CYFRAME) * 2 + Api_GetSystemMetrics(SM_CYSIZE)

    Ret = Api_SetRect(rct, 5, 5, GetWidth - fWidth - 5, GetHeight - fHeight - 5)
    hBrush = Api_CreateHatchBrush(HS_DIAGCROSS, vbBlue)  'VBの場合青のDIAGCROSSで描画される
```

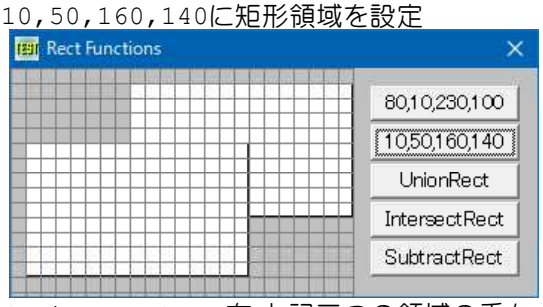
```
    Ret = Api_FrameRect(hfDC, rct, hBrush)

    Ret = Api_DeleteObject(hWhiteBrush)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
    Ret = Api_ReleaseDC(GethWnd, hfDC)
    Ret = Api_ReleaseDC(Picture1.GethWnd, hpDC)
End Sub

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

矩形領域描画のいろいろ

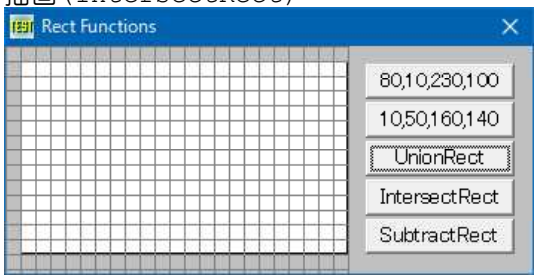
矩形領域の描画の方法数題
Rectangle　長方形の描画
SetRect　RECT構造体の値を設定
UnionRect　2つの矩形を内包する矩形サイズを取得
IntersectRect　2つの矩形が重なっているとき、重なっている部分の座標値を返す
SubtractRect　一つの矩形から別の矩形を差し引いた矩形を作成

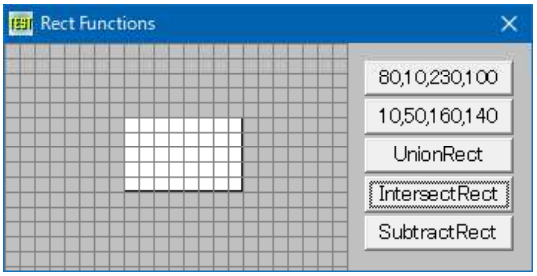
左:80,10,230,100に矩形領域を設定　　右:10,50,160,140に矩形領域を設定



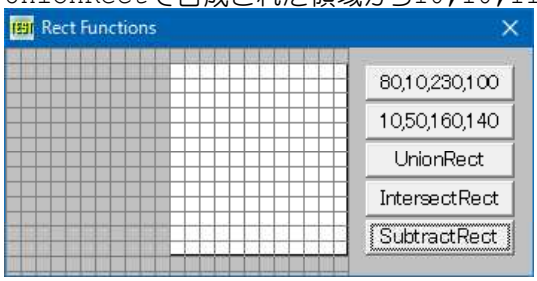


左:上記二つの矩形領域を内包する矩形を描画(UnionRect)　　右:上記二つの領域の重なっている部分の領域を描画(IntersectRect)





UnionRectで合成された領域から10,10,110,140の矩形領域部分を差し引いた矩形領域を描画



座標確認のため10ドット間隔でグリッドを表示させています。

```
'====================================
'= 矩形領域描画のいろいろ
'=    (Rectangle.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&)

' 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&)

' 2つの矩形を内包する矩形サイズを取得
Declare Function Api_UnionRect& Lib "user32" Alias "UnionRect" (lpDestRect As RECT,
lpSrc1Rect As RECT, lpSrc2Rect As RECT)

' 2つの矩形が重なっているとき、重なっている部分の座標値を返す
Declare Function Api_IntersectRect& Lib "user32" Alias "IntersectRect" (lpDestRect As
RECT, lpSrc1Rect As RECT, lpSrc2Rect As RECT)

' 一つの矩形から別の矩形を差し引いた矩形を作成
Declare Function Api_SubtractRect& Lib "user32" Alias "SubtractRect" (lprcDst As RECT,
lprcSrc1 As RECT, lprcSrc2 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&)

Var Shared Button(4) As Object

For i = 0 To 4
    Button(i).Attach GetDlgItem("Button" & Trim$(Str$(i + 1)))
    Button(i).SetFontSize 14
Next

Var Shared Src1 As RECT
Var Shared Src2 As RECT
Var Shared URect As RECT
Var Shared SRect As RECT
Var Shared FLG As Byte
Var Shared hDC As Long

'====================================
'=
'====================================
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    hDC = Api_GetDC(GethWnd)
End Sub

'====================================
'=
'====================================
Declare Sub DrawGrid edecl ()
Sub DrawGrid()
    For x = 0 To 230 Step 10
        Line(x, 0) - (x, 160), , 1
    Next
```

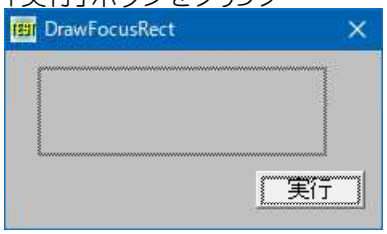
```
    For y = 0 To 160 Step 10
        Line(0, y) - (230, y), , 1
    Next
End Sub

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

    Cls
    Ret = Api_SetRect(Src1, 80, 10, 230, 100)
    Ret = Api_Rectangle(hDC, Src1.Left, Src1.Top, Src1.Right, Src1.Bottom)
    DrawGrid
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var Ret As Long
    Ret = Api_SetRect(Src2, 10, 50, 160, 140)
    Ret = Api_Rectangle(hDC, Src2.Left, Src2.Top, Src2.Right, Src2.Bottom)

    DrawGrid
End Sub

'=====================================
'=
'=====================================
Declare Sub Button3_on edecl ()
Sub Button3_on()
    Var Ret As Long

    Cls
    Ret = Api_UnionRect(URect, Src1, Src2)
    Ret = Api_Rectangle(hDC, URect.Left, URect.Top, URect.Right, URect.Bottom)

    DrawGrid
End Sub

'=====================================
'=
'=====================================
Declare Sub Button4_on edecl ()
Sub Button4_on()
    Var Ret As Long

    Cls

    Ret = Api_IntersectRect(URect, Src1, Src2)
    Ret = Api_Rectangle(hDC, URect.Left, URect.Top, URect.Right, URect.Bottom)

    DrawGrid
End Sub

'=====================================
'=
'=====================================
Declare Sub Button5_on edecl ()
Sub Button5_on()
    Var Ret As Long

    Cls

    Ret = Api_SetRect(SRect, 10, 10, 110, 140)
    Ret = Api_SubtractRect(URect, URect, SRect)
```

```
    Ret = Api_Rectangle(hDC, URect.Left, URect.Top, URect.Right, URect.Bottom)

    DrawGrid
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_QueryClose edecl (Cancel%,ByVal Mode%)
Sub MainForm_QueryClose(Cancel%,ByVal Mode%)
    If Cancel% = 0 Then
        Ret = Api_ReleaseDC(GethWnd, hDC)
        End
    End If
End Sub
'=====================================
'=
'=====================================
While 1
    WaitEvent
Wend
Stop
End
```

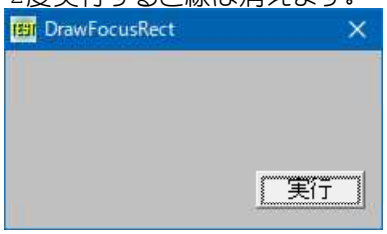
矩形領域枠を点線で描画する

矩形領域枠を点線で描画します。
DrawFocusRect 指定の矩形の周囲にフォーカスを示す点線を描画する

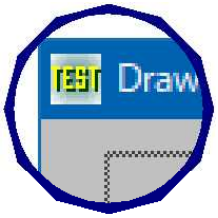
「実行」ボタンをクリック



2度実行すると線は消えます。



点線であることの確認（3倍に拡大）

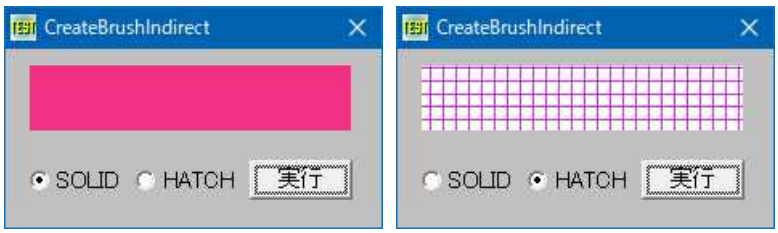



```
'=====================================
'= 矩形領域枠を点線で描画する
'=    (DrawFocusRect.bas)
'=====================================
#include "Windows.bi"

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

' 指定の矩形の周囲にフォーカスを示す点線を描画する
Declare Function Api_DrawFocusRect& Lib "user32" Alias "DrawFocusRect" (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&)

Var Shared hDC As Long

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

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

    hDC = Api_GetDC(GethWnd)

    rct.Top = 10
    rct.Left = 20
    rct.Right = 200
    rct.Bottom = 65

    Ret = Api_DrawFocusRect(hDC, rct)
    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

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

    Ret = Api_ReleaseDC(GethWnd, hDC)
    End
End Sub

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

矩形領域を塗る

指定した矩形領域を純色(SOLID)、ハッチ(HATCHED)で塗るテスト
CreateBrushIndirect ブラシを作成する関数
CreateRectRgn 長方形のリージョンを作成
IsWindow ウィンドウハンドルが有効か調べる
SetRectEmpty 空の長方形を作成する
SetRect オブジェクトの矩形を設定
IsRectEmpty 指定された長方形が空であるかどうかを判断
IntersectRect 矩形が重なっているか調べる
RedrawWindow 領域内の指定の範囲を再描画
GetRgnBox リージョンの境界長方形の取得
DeleteObject オブジェクトの削除
FillRect 四角形を塗りつぶす
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




```
'=======================================
'= 矩形領域を塗る
'=    (CreateBrushIndirect.bas)
'=======================================
#include "Windows.bi"
```

```
Type LOGBRUSH
    lbStyle    As Long
    lbColor    As Long
    lbHatch    As Long
End Type

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

' LOGBRUSH構造体を定義して論理ブラシを作成
Declare Function Api_CreateBrushIndirect& Lib "gdi32" Alias "CreateBrushIndirect"
(lpLogBrush As LOGBRUSH)

' 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&)

' 指定されたウィンドウ ハンドルが既存のウィンドウを識別しているどうかを判断
Declare Function Api_IsWindow& Lib "user32" Alias "IsWindow" (ByVal hWnd&)

' RECT構造体の値を全て0にする
Declare Function Api_SetRectEmpty& Lib "user32" Alias "SetRectEmpty" (lpRect As RECT)

' 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&)

' CRectが空かどうかを調べる
Declare Function Api_IsRectEmpty& Lib "user32" Alias "IsRectEmpty" (lpRect As RECT)

' 2つの矩形が重なっているとき、重なっている部分の座標値を返す
Declare Function Api_IntersectRect& Lib "user32" Alias "IntersectRect" (lpDestRect As
RECT, lpSrc1Rect As RECT, lpSrc2Rect As RECT)

' リージョンの境界長方形の取得
Declare Function Api_GetRgnBox& Lib "gdi32" Alias "GetRgnBox" (ByVal hRgn&, lpRect As
RECT)

' 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&)

' 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&)

' 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 RDW_INVALIDATE &H1                  '再描画領域を無効化
#define BS_SOLID 0                          'ソリッドブラシ
#define BS_HATCHED 2                        'ハッチング(スタイルはlbHatchで指定)
#define HS_CROSS 4                          '水平と垂直クロスハッチ

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 Sub Button1_on edecl ()
Sub Button1_on()
    Var lb As LOGBRUSH
    Var rc As RECT
    Var RgnRect As RECT
    Var Rgn As Long
    Var hBrush As Long
    Var hDC As Long
    Var Ret As Long

    hDC = Api_GetDC(GethWnd)

    Randomize Val(Right$(Time$, 1))
    lb.lbColor = RGB(Int(Rnd * 256), Int(Rnd * 256), Int(Rnd * 256))

    If Radio1.GetCheck = 1 Then
        lb.lbStyle = BS_SOLID
    Else
        lb.lbStyle = BS_HATCHED
    End If

    lb.lbHatch = HS_CROSS
    hBrush = Api_CreateBrushIndirect(lb)
    Ret = Api_SetRect(rc, 0, 0, 220, 60)
    Rgn = Api_CreateRectRgn(15, 10, 215, 50)
    Ret = Api_GetRgnBox(Rgn, RgnRect)
    Ret = Api_IntersectRect(rc, RgnRect, rc)
    Ret = Api_SetRectEmpty(RgnRect)
    Ret = Api_FillRect(hDC, rc, hBrush)
    Ret = Api_DeleteObject(hBrush)
    Ret = Api_ReleaseDC(GethWnd, hDC)
    Ret = Api_IsRectEmpty(RgnRect)
    If Ret <> 0 Then Ret = Api_SetRectEmpty(RgnRect)
End Sub

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

クライアント領域の表示状態を調べる

クライアント領域の表示状態を調べます。
FindWindow クラス名、キャプション名を与えてウィンドウハンドルを取得
GetClipBox デバイスコンテキストの歌詞領域を描画できる最小境界長方形の寸法を取得
GetClientRect ウィンドウのクライアント領域座標を取得
EqualRect 二つの長方形の寸法を比較して、等しいかどうかを調べる
GetDC 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

前もってメモ帳を起動しておき、その領域の表示状態を調べています。

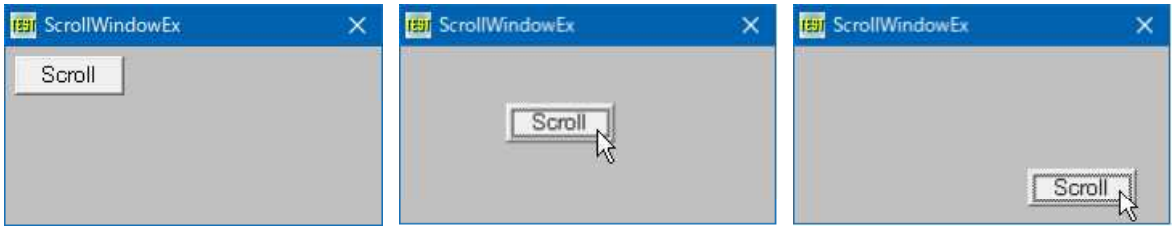
注）
Windows 7 において、Aeroテーマではどの状態でも「完全に見えている」となります。
Windows Vista では、試しておりませんが同様ではないかと‥

```
'=====================================
'= クライアント領域の表示状態を調べる
'=    (EqualRect.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$,
ByVal lpWindowName$)

' デバイスコンテキストの可視領域を描画できる最小境界長方形の寸法を取得
Declare Function Api_GetClipBox& Lib "gdi32" Alias "GetClipBox" (ByVal hDC&, lprc As
RECT)
' 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)

' 2つの長方形の寸法を比較して、長方形が等しいかどうかを判断
Declare Function Api_EqualRect& Lib "user32" Alias "EqualRect" (lprc1 As RECT, lprc2 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&)

#define ERROR 0                                 'エラー
#define NULLREGION 1                            '空のリージョンが作成された
#define SIMPLEREGION 2                          '境界の重ならないリージョンが作成された
#define COMPLEXREGION 3                         'ひとつ以上の長方形からなるリージョンが作成された

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 ClassName As String
    Var hWindow As Long
    Var hDC As Long
    Var ClientRect As RECT
    Var ClipBoxRect As RECT
    Var Ret As Long
```

```
    'クラス名でウィンドウハンドルを取得
    ClassName = "Notepad"

    hWindow = Api_FindWindow(ClassName, ByVal 0)

    'ウィンドウハンドルが取得できた場合は
    If hWindow <> 0 Then

        'クライアント領域に対応するデバイスコンテキストのハンドルを取得
        hDC = Api_GetDC(hWindow)

        '可視領域を描画できる最小境界長方形の寸法を取得
        Select Case Api_GetClipBox(hDC, ClipBoxRect)

            'リージョンは空
            Case NULLREGION
                Text1.SetWindowText "完全に隠れている"

            'リージョンは単一の長方形
            Case SIMPLEREGION

                'ウィンドウのクライアント領域の寸法を取得
                Ret = Api_GetClientRect(hWindow, ClientRect)

                '可視領域を描画できる最小境界長方形の寸法と比較
                If Api_EqualRect(ClipBoxRect, ClientRect) Then
                    Text1.SetWindowText "完全に見えている"
                Else
                    Text1.SetWindowText "単一の長方形に見えるように隠れている"
                End If

            'リージョンは複数の長方形
            Case COMPLEXREGION
                Text1.SetWindowText "複数の長方形に見えるように隠れている"

            'エラー
            Case ERROR
                Text1.SetWindowText "エラーで取得できない"
        End Select

        'デバイスコンテキストを解放
        Ret = Api_ReleaseDC(hWindow, hDC)
    Else
        A% = MessageBox("EqualRect", "メモ帳を起動してください！", 0, 2)
    End If
End Sub

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

クライアント領域をスクロール

例では、コマンドボタンをクリックするたびにスクロール（移動）させています。
ScrollWindowEx 指定したウィンドウのクライアント領域の内容をスクロール
SW_ERASE(&H4) ウィンドウを隠す
SW_INVALIDATE(&H2) スクロール後に無効化
SW_SCROLLCHILDREN(&H1) すべての子ウィンドウをスクロール




```
'=====================================
'= クライアント領域をスクロール
'=    (ScrollWindowEx.bas)
'=====================================
#include "Windows.bi"

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

' 指定したウィンドウのクライアント領域の内容をスクロール
Declare Function Api_ScrollWindowEx& Lib "user32" Alias "ScrollWindowEx" (ByVal hWnd&,
ByVal dx&, ByVal dy&, prcScroll As RECT, prcClip As RECT, ByVal hrgnUpdate&, prcUpdate As
RECT, ByVal Flags&)

#define SW_ERASE &H4                            'ウィンドウを隠す
#define SW_INVALIDATE &H2                       'スクロール後に無効化
#define SW_SCROLLCHILDREN &H1                   'すべての子ウィンドウをスクロール

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

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var Scrl As RECT
    Var Clip As RECT
    Var hUpdate As Long
    Var pUpdate As RECT
    Var Ret As Long

    Scrl.Top = 0
    Scrl.Left = 0
    Scrl.Right = GetWidth - Button1.GetWidth
    Scrl.Bottom = GetHeight - Button1.GetHeight

    'クリック毎にx=20、y=10ずつスクロール
    Ret = Api_ScrollWindowEx(GethWnd, 20, 10, Scrl, Clip, hUpdate, pUpdate,
SW_SCROLLCHILDREN)
    Refresh
    SetFocus
End Sub

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

クラス名でアプリの起動の確認と終了

あらかじめ知られているアプリケーションのクラス名を指定しておきます。
EXCEL→XLMAIN　WORD→OpusApp　メモ帳→Notepad　関数電卓→SciCalc
ラジオボタンをチェックしボタンをクリックします。
FindWindow ウインドウハンドル取得
SendMessage 指定ウィンドウにメッセージを送る
GetWindowRect サイズ等を取得する

起動されている場合そのフォーム幅、高さを表示します。

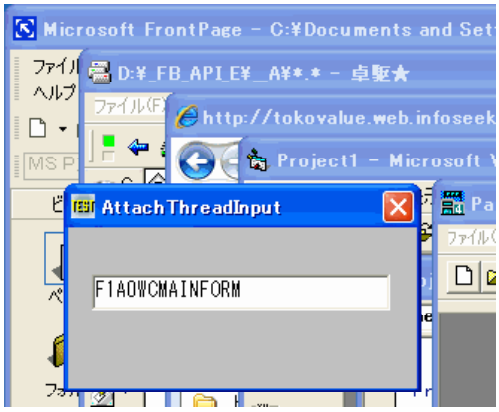

```
'=======================================
'= クラス名で起動の確認と終了
'=    (FindWindow4.bas)
'=======================================
#include "Windows.bi"

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)

' 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 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 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 Radio(3) As Object
Var Shared Text(1) As Object
Var Shared Button(2) As Object

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

Var Shared rd(3) As String
Var Shared Flg As byte
Var Shared hWnd As Long
Var Shared FileName As String

'=======================================
'=  クラス名で起動の確認と終了
'=======================================
```

```
Declare Sub MainForm_Start edecl ()
Sub MainForm_Start()
    rd(0) = "XLMAIN"
    rd(1) = "OpusApp"
    rd(2) = "Notepad"
    rd(3) = "SciCalc"

    Flg = 0
    FileName = "Excel.exe"
    Button(1).EnableWindow 0
End Sub

'=====================================
'=  クラス名で起動の確認と終了
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var rct As RECT
    Var Ret As Long

    'クラス名を与えてAPPのハンドルを取得
    '起動中ならハンドルが返り、起動していなければ0が返る
    hWnd = Api_FindWindow(rd(Flg), vbNullString)
    Ret = Api_GetWindowRect(hWnd, rct)

    If hWnd = 0 Then
        Text(1).ShowWindow 0
        Text(0).SetWindowText "【" & rd(Flg) & "】は起動されていません。"
        Button(0).EnableWindow -1
        Button(1).EnableWindow 0
    Else
        Text(1).ShowWindow -1
        Text(0).SetWindowText "【" & rd(Flg) & "】は起動されています。"
        Text(1).SetWindowText "FormWidth=" & Trim$(Str$(rct.Right - rct.Left)) & " /
FormHeight=" & Trim$(Str$(rct.Bottom - rct.Top))

        Button(0).EnableWindow 0
        Button(1).EnableWindow -1
    End If
End Sub

'=====================================
'=  確認と終了
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()

    'エクセル(アプリ)が起動中なら終了する場合
    '指定のハンドルに終了のメッセージを送る
    Ret = Api_SendMessage(hWnd, WM_CLOSE, 0, 0)
    Text(0).SetWindowText "【" & rd(Flg) & "】は終了しました。"
    Text(1).SetWindowText ""
    Wait 100
    Button(0).EnableWindow -1
    Button(1).EnableWindow 0
    Text(0).SetWindowText ""
End Sub

'=====================================
'=  起動
'=====================================
Declare Sub Button3_on edecl ()
Sub Button3_on()
    Shell FileName, , 5
End Sub

'=====================================
'=
'=====================================
Declare Sub Radio1_on edecl ()
```

```
Sub Radio1_on()
    Flg = 0
    FileName = "Excel.exe"
End Sub

Declare Sub Radio2_on edecl ()
Sub Radio2_on()
    Flg = 1
    FileName  = "Winword.exe"
End Sub

Declare Sub Radio3_on edecl ()
Sub Radio3_on()
    Flg = 2
    FileName  = "Notepad.exe"
End Sub

Declare Sub Radio4_on edecl ()
Sub Radio4_on()
    Flg = 3
    FileName  = "Calc.exe"
End Sub

'=====================================
'=  クラス名で起動の確認と終了
'=====================================
While 1
    WaitEvent
Wend
STop
End
```

クラス名の取得

GetClassName ウィンドウのクラス名を取得
GetFocus フォーカスを持つウィンドウハンドルを取得
GetForegroundWindow ユーザーが操作中のウインドウを取得
GetWindowThreadProcessId ウィンドウのプロセスIDとスレッドIDを取得
SendMessage ウィンドウにメッセージを送信
AttachThreadInput 特定のスレッドの入力処理機構を別のスレッドにアタッチ
SetWindowPos ウィンドウのサイズ、位置、および Z オーダーを設定



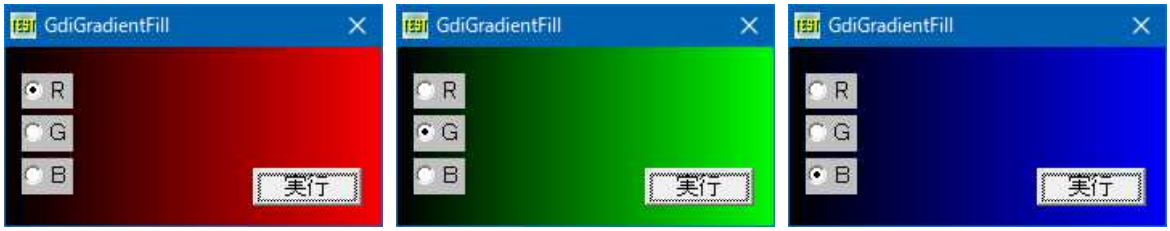

```
'=====================================
'= クラス名の取得
'=    (AttachThreadInput.bas)
'=====================================
#include "Windows.bi"

' 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&)
```

```
' フォーカスを持つウィンドウハンドルを取得
Declare Function Api_GetFocus& Lib "user32" Alias "GetFocus" ()

' 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&)

' 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)

' 特定のスレッドの入力処理機構を別のスレッドにアタッチ
Declare Function Api_AttachThreadInput& Lib "user32" Alias "AttachThreadInput" (ByVal
idAttach&, ByVal idAttachTo&, ByVal fAttach&)

' 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 HWND_BOTTOM 1                          'ウィンドウを最背面に配置
#define HWND_NOTOPMOST (-2)                    'ウィンドウを常に最前面に配置(他のウィンドウが
                                               HWND_TOPMOSTに配置されている場合はその配下)
#define HWND_TOP 0                             'ウィンドウを最前面に配置
#define HWND_TOPMOST (-1)                      'ウィンドウを常に最前面に配置

#define SWP_DRAWFRAME &H20                     '再描画のときウィンドウを囲む枠も描画
#define SWP_FRAMECHANGED &H20                  'ウィンドウサイズ変更中でなくてもWM_NCCALCSIZEを送る
#define SWP_HIDEWINDOW &H80                    'ウィンドウを隠す
#define SWP_NOACTIVATE &H10                    'ウィンドウをアクティブにしない
#define SWP_NOCOPYBITS &H100                   'クライアント領域の内容をクリアする
#define SWP_NOMOVE &H2                         'ウィンドウの現在位置を保持する
#define SWP_NOOWNERZORDER &H200                'オーナーウィンドウのZオーダーは変えない
#define SWP_NOREDRAW &H8                       'ウィンドウを自動的に再描画しない
#define SWP_NOREPOSITION &H200                 'SWP_NOOWNERZORDERと同じ
#define SWP_NOSIZE &H1                         'ウィンドウの現在のサイズを保持する
#define SWP_NOZORDER &H4                       'ウィンドウリスト内での現在位置を保持する
#define SWP_SHOWWINDOW &H40                    'ウィンドウを表示する

Var Shared hWndCtrl As Long
Var Shared Timer1 As Object
Var Shared Edit1 As Object
Timer1.Attach GetDlgItem("Timer1")
Edit1.Attach GetDlgItem("Edit1") : Edit1.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub GetControl()
Sub GetControl()
    Var Buffer As String
    Var CurrentThreadID As Long
    Var OtherThreadID As Long
    Var Thread1 As Long
    Var Thread2 As Long
    Var bMerker As Integer
    Var Ret As Long

    'カレントスレッド取得
    Thread1 = Api_GetWindowThreadProcessId(GethWnd, CurrentThreadID)

    'ユーザーが操作中のウインドウを取得
    Thread2 = Api_GetWindowThreadProcessId(Api_GetForegroundWindow, OtherThreadID)

    'スレッドをAttach
    If Thread1 <> Thread2 Then bMerker = Api_AttachThreadInput(Thread2, Thread1, True)

    Ret = Api_GetFocus()
```

```
    hWndCtrl = Ret
    Edit1.SetWindowText Str$(Ret)

    'スレッドをDetach
    If bMerker Then bMerker = Api_AttachThreadInput(Thread2, Thread1, False)

    'クラス名取得
    Buffer$ = Space$(255)
    Ret = Api_GetClassName(hWndCtrl, Buffer, Len(Buffer))
    Edit1.SetWindowText Buffer
End Sub

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

    '自身を最前面に
    Ret = Api_SetWindowPos(GethWnd, HWND_TOPMOST, 0, 0, 0, 0, SWP_NOSIZE Or SWP_NOMOVE)

    Timer1.SetInterval 50
    Timer1.Enable -1
End Sub

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

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

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

グラデーション塗り潰し(Ⅰ)

フォームをグラデーションで塗り潰します。
GdiGradientFill 長方形と三角形構造内を塗り潰す
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



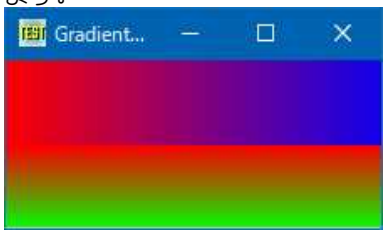
```
'=====================================
'= グラデーション塗り潰し
'=    (GdiGradientFill.bas)
'=====================================
#include "Windows.bi"

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

#define GRADIENT_FILL_RECT_H &H0            '水平方向にグラデーション
#define GRADIENT_FILL_RECT_V &H1            '垂直方向にグラデーション
#define GRADIENT_FILL_TRIANGLE &H2          '三角形グラデーション
#define GRADIENT_FILL_OP_FLAG &HFF

' 長方形と三角形構造内を塗り潰す
Declare Function Api_GdiGradientFill& Lib "gdi32" Alias "GdiGradientFill" (ByVal hDC&,
pVertex As TRIVERTEX, ByVal dwNumVertex&, pMesh As GRADIENT_RECT, ByVal dwNumMesh&, ByVal
dwMode&)

' 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&)

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

'=====================================
'=
'=====================================
Declare Function LongToUShort(Unsigned As Long) As Integer
Function LongToUShort(Unsigned As Long) As Integer
    LongToUShort = int(Unsigned - &H10000)
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var vert(1) As TRIVERTEX
    Var gRect As GRADIENT_RECT
    Var hDC As Long
    Var Ret As Long

    hDC = Api_GetDC(GethWnd)

    '黒～
    vert(0).x = 0
    vert(0).y = 0
    vert(0).Red = 0
    vert(0).Green = 0
```

```
    vert(0).Blue = 0
    vert(0).Alpha = 0

    '～赤･緑･青
    Select Case Index
    Case 0
        vert(1).Red = LongToUShort(&HFF00)
        vert(1).Green = 0
        vert(1).Blue = 0
    Case 1
        vert(1).Red = 0
        vert(1).Green = LongToUShort(&HFF00)
        vert(1).Blue = 0
    Case 2
        vert(1).Red = 0
        vert(1).Green = 0
        vert(1).Blue = LongToUShort(&HFF00)
    End Select

    vert(1).x = GetWidth
    vert(1).y = GetHeight
    vert(1).Alpha = 0

    gRect.UpperLeft = 0
    gRect.LowerRight = 1
    Ret = Api_GdiGradientFill(hDC, vert(0), 2, gRect, 1, GRADIENT_FILL_RECT_H)

    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

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

グラデーションで塗り潰し(II)

長方形又は三角形領域をグラデーションで塗りつぶします。
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
GradientFill 長方形又は三角形をグラデーションで塗りつぶす
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

上半分を水平方向に、下半分を縦方向にグラデーションで塗り潰しています。
WindowsXPとWindows2000では、メニューバーの高さが異なるためクライアントエリアの高さを求めて2等分しています。




```
'=====================================
'= 長方形又は三角形領域をグラデーションで塗りつぶす
'=    (GradientFillRect.bas)
'=====================================
#include "Windows.bi"

Type RECT
    Left       As Long
```

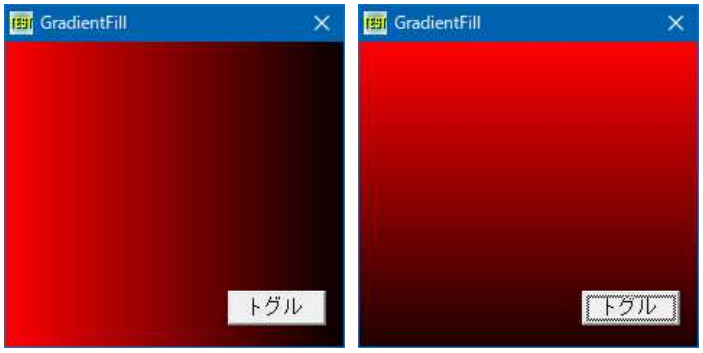
```
    Top         As Long
    Right       As Long
    Bottom      As Long
End Type

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

' 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&)

' 長方形又は三角形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_RECT, ByVal dwNumMesh&, ByVal
dwMode&)

' 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 SM_CYMIN 29                         'ウィンドウの最小高さ
#define SM_CYBORDER 6                       'サイズ固定ウィンドウの境界線のY方向の幅
#define GRADIENT_FILL_RECT_H &H0
#define GRADIENT_FILL_RECT_V &H1
#define GRADIENT_FILL_TRIANGLE &H2
#define GRADIENT_FILL_OP_FLAG &HFF

'=====================================
'=
'=====================================
Declare Function CCol(ByVal Col As Byte) As Integer
Function CCol(ByVal Col As Byte) As Integer
    If Col > &H7F Then
        CCol = (Col * &H100) - &H10000
    Else
        CCol = Col * &H100
    End If
End Function

'=====================================
'=
'=====================================
Declare Sub DrawGradient()
Sub DrawGradient()
    Var hDC As Long
    Var fHeight As Long
    Var Vertex(1) As TRIVERTEX
    Var gr As GRADIENT_RECT
    Var Ret As Long

    Cls

    hDC = Api_GetDC(GethWnd)

    '枠の高さ
    fHeight = Api_GetSystemMetrics(SM_CYMIN) + Api_GetSystemMetrics(SM_CYBORDER)
```

```
    '上半分横方向グラデーション
    Vertex(0).x = 0
    Vertex(0).y = 0
    Vertex(0).Red = CCol(255)
    Vertex(0).Green = 0
    Vertex(0).Blue = 0
    Vertex(0).Alpha = 0

    Vertex(1).x = GetWidth
    Vertex(1).y = (GetHeight - fHeight) / 2
    Vertex(1).Red = 0
    Vertex(1).Green = 0
    Vertex(1).Blue = CCol(255)
    Vertex(1).Alpha = 0

    gr.UpperLeft = 0
    gr.LowerRight = 1
    Ret = Api_GradientFill(hDC, Vertex(0), 2, gr, 1, GRADIENT_FILL_RECT_H)

    '下半分縦方向グラデーション
    Vertex(0).x = 0
    Vertex(0).y = (GetHeight - fHeight)  / 2
    Vertex(0).Red = CCol(255)
    Vertex(0).Green = 0
    Vertex(0).Blue = 0
    Vertex(0).Alpha = 0

    Vertex(1).x = GetWidth
    Vertex(1).y = (GetHeight - fHeight)
    Vertex(1).Red = 0
    Vertex(1).Green = CCol(255)
    Vertex(1).Blue = 0
    Vertex(1).Alpha = 0
    gr.UpperLeft = 0
    gr.LowerRight = 1
    Ret = Api_GradientFill(hDC, Vertex(0), 2, gr, 1, GRADIENT_FILL_RECT_V)
End Sub

'=====================================
'=
'=====================================
Declare Sub MainForm_Resize edecl ()
Sub MainForm_Resize()
    DrawGradient
End SUb

'=====================================
'=
'=====================================
Declare Sub MainForm_QueryClose edecl ()
Sub MainForm_QueryClose()
    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

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

グラデーションで塗り潰し(III)

長方形又は三角形領域をグラデーションで塗りつぶします。
GradientFill 長方形又は三角形をグラデーションで塗りつぶす

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




```
'=======================================
'= グラデーションで塗りつぶす
'=    (GradientFill.bas)
'=======================================
#include "Windows.bi"
#define GRADIENT_FILL_RECT_H &H0            '水平方向にグラデーション
#define GRADIENT_FILL_RECT_V &H1            '垂直方向にグラデーション
#define GRADIENT_FILL_TRIANGLE &H2          '三角形グラデーション
#define vbRed &H0000FF                      '赤のカラーコード
#define vbBlack &H000000                    '黒のカラーコード

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 Any, ByVal dwNumVertex&, pMesh As Any, ByVal dwNumMesh&, ByVal dwMode&)

' 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 GraFillDirection As Long
Var Shared hDC As long

Var Shared Button1 As Object

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

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

    hDC = Api_GetDC(GethWnd)
End Sub

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

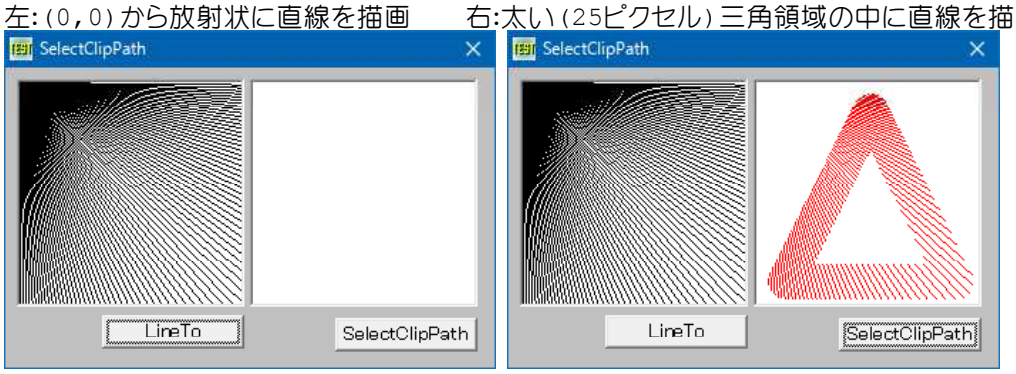
```
Declare Function LongToShort(Unsigned As Long) As Integer
Function LongToShort(Unsigned As Long) As Integer
    If Unsigned < 32768 Then
       LongToShort = CInt(Unsigned)
    Else
       LongToShort = CInt(Unsigned - &H10000)
    End If
End Function

'=====================================
'=
'=====================================
Declare Sub DrawGradientFill(ByVal Col1 As Long, ByVal Col2 As Long)
Sub DrawGradientFill(ByVal Col1 As Long, ByVal Col2 As Long)
    Var vert(1) As TRIVERTEX
    Var grc As GRADIENT_RECT

   '左上
    vert(0).x = 0
    vert(0).y = 0
    vert(0).Red = LongToShort((Col1 And &HFF) * 256)
    vert(0).Green = LongToShort(((Col1 And &HFF00) ¥ &H100) * 256)
    vert(0).Blue = LongToShort(((Col1 And &HFF0000) ¥ &H10000) * 256)
    vert(0).Alpha = 0

   '右下
    vert(1).x = GetWidth
    vert(1).y = GetHeight
    vert(1).Red = LongToShort((Col2 And &HFF) * 256)
    vert(1).Green = LongToShort(((Col2 And &HFF00) ¥ &H100) * 256)
    vert(1).Blue = LongToShort(((Col2 And &HFF0000) ¥ &H10000) * 256)
    vert(1).Alpha = 0
    grc.LowerRight = 0
    grc.UpperLeft = 1

    Cls

    Ret = Api_GradientFill(hDC, vert(0), 2, grc, 1, Abs(GraFillDirection))
End Sub

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

   '切替
    GraFillDirection = Not GraFillDirection
    DrawGradientFill vbRed, vbBlack
End Sub

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

    DrawGradientFill vbRed, vbBlack
End Sub

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

    Ret = Api_ReleaseDC(gethWnd, hDC)
```

```
End Sub

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

クリッピング領域とパスを結合(Ⅰ)

SetPolyFillMode 多角形塗りつぶしモードを設定
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
BeginPath 指定されたデバイスコンテキストのパスを指定
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
CloseFigure パス内で現在開いている図形を閉じる
EndPath パスの作成を終了
SelectClipPath 指定したデバイスコンテキストのクリッピング領域とパスを結合
WidenPath パスを塗りつぶしの対象領域として再定義
DeleteObject システムリソースの解放
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ハンドルを解放

左:(0,0)から放射状に直線を描画　　右:太い(25ピクセル)三角領域の中に直線を描画




```
'=====================================
'= クリッピング領域とパスを結合
'=    (SelectClipPath.bas)
'=====================================
#include "Windows.bi"

' 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&)

' 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&)

' 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&)

' 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&)
```

```
' 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&)

' パス内で現在開いている図形を閉じる
Declare Function Api_CloseFigure& Lib "gdi32" Alias "CloseFigure" (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&)

' 指定したデバイスコンテキストのクリッピング領域とパスを結合
Declare Function Api_SelectClipPath& Lib "gdi32" Alias "SelectClipPath" (ByVal hDC&,
ByVal iMode&)

' 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&)

' 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 WINDING 2                                   '全域塗りつぶしモード
#define RGN_COPY 5                                  'HRGNSRC1のコピーを作成
#define PS_SOLID 0                                  '実線のペンを作成
#define vbWhite &HFFFFFF                            '白のカラーコード
#define vbBlack &H000000                            '黒のカラーコード
#define vbRed &H0000FF                              '赤のカラーコード
Var Shared Picture1 As Object
Var Shared Picture2 As Object
Var Shared Button1 As Object
Var Shared Button2 As Object

Picture1.Attach GetDlgItem("Picture1")
Picture2.Attach GetDlgItem("Picture2")
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Button2.Attach GetDlgItem("Button2") : Button2.SetFontSize 14

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hDC1 As Long
    Var hPen As Long
    Var hOldPen As Long
    Var i As Long
    Var Ret As Long

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

    '多角形塗りつぶしモードを設定
    Ret = Api_SetPolyFillMode(hDC1, WINDING)

    '論理ペンを作成
    hPen = Api_CreatePen(PS_SOLID, 1, vbBlack)

    'オブジェクトを選択
    hOldPen = Api_SelectObject(hDC1, hPen)

    For i = 1 To 400 Step 5

        '座標(0,0)から直線を描画
        Ret = Api_MoveToEx(hDC1, 0, 0, 0)
        Ret = Api_LineTo(hDC1, i, 400 - i)
    Next
```

```
    '論理オブジェクトを削除
    Ret = Api_DeleteObject(hPen)

    'デバイスコンテキストを解放
    Ret = Api_ReleaseDC(Picture1.GethWnd, hDC1)
End Sub

'=====================================
'=
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var hDC2 As Long
    Var hPen As Long
    Var hOldPen As Long
    Var i As Long
    Var Ret As Long

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

    Ret = Api_SetPolyFillMode(hDC2, WINDING)

    'パス用ペンを設定
    hPen = Api_CreatePen(PS_SOLID, 25, vbWhite)
    hOldPen = Api_SelectObject(hDC2, hPen)

    'パスブラケットをオープン
    Ret = Api_BeginPath(hDC2)

    'Pictureに三角形を描画
    Ret = Api_MoveToEx(hDC2, 80, 20, 0)      '三角形頂点      ・
    Ret = Api_LineTo(hDC2, 20, 140)          '底辺左へ        ／
    Ret = Api_LineTo(hDC2, 140, 140)         '底辺右へ        ―
    Ret = Api_CloseFigure(hDC2)              '図形を閉じる    ＼

    'パスブラケットをクローズ
    Ret = Api_EndPath(hDC2)

    '三角形をクリッピングパスとして設定
    Ret = Api_WidenPath(hDC2)

    'クリッピング領域とパスを結合
    Ret = Api_SelectClipPath(hDC2, RGN_COPY)

    'オブジェクトを選択
    Ret = Api_SelectObject(hDC2, hOldPen)

    'パス用ペンを設定
    hPen = Api_CreatePen(PS_SOLID, 1, vbRed)
    hOldPen = Api_SelectObject(hDC2, hPen)

    For i = 1 To 400 Step 5

        '座標(0,0)から直線を描画
        Ret = Api_MoveToEx(hDC2, 0, 0, 0)
        Ret = Api_LineTo(hDC2, i, 400 - i)
    Next

    '論理オブジェクトを削除
    Ret = Api_DeleteObject(hPen)

    'デバイスコンテキストを解放
    Ret = Api_ReleaseDC(Picture2.GethWnd, hDC2)
End Sub

'=====================================
'=
'=====================================
While 1
```

```
    WaitEvent
Wend
Stop
End
```

クリッピング領域とパスを結合（II）

CreateFontIndirect 論理フォントを作成
TextOut テキストを指定の位置に出力
SetPolyFillMode 多角形塗りつぶしモードを設定
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
BeginPath 指定されたデバイスコンテキストのパスを指定
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
EndPath パスの作成を終了
SelectClipPath 指定したデバイスコンテキストのクリッピング領域とパスを結合
WidenPath パスを塗りつぶしの対象領域として再定義
SetBkMode バックグラウンドの塗りつぶしモード設定
DeleteObject システムリソースの解放
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ハンドルを解放


```
'=======================================
'= クリッピング領域とパスを結合（II）
'=    (SelectClipPath2.bas)
'=======================================
#include "Windows.bi"

#define LF_FACESIZE 32

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

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)
```

```
' パスをクリッピング領域に設定
Declare Function Api_SelectClipPath& Lib "gdi32" Alias "SelectClipPath" (ByVal hDC&,
ByVal iMode&)

' 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&)

' 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&)

' 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&)

' 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&)

' 論理ペンを作成
Declare Function Api_CreatePen& Lib "gdi32" Alias "CreatePen" (ByVal nPenStyle&, ByVal
nWidth&, ByVal crColor&)

' 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&)

' 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&)

' 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&)

' 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
#define RGN_COPY 5                       'HRGNSRC1のコピーを作成
#define WINDING 2                        '全域塗りつぶしモード
#define PS_SOLID 0                       '実線のペンを作成
#define vbWhite &HFFFFFF                 '白のカラーコード
#define vbBlack &H000000                 '黒のカラーコード
#define vbRed &H0000FF                   '赤のカラーコード
#define DT_CALCRECT &H400                'テキストを表示するために必要な長方形の大きさを
                                         lpRectパラメータに格納
Var Shared Button1 As Object

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

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var rc As RECT
    Var lf As LOGFONT
```

```
    Var rFont As Long
    Var hDC As Long
    Var hPen As Long
    Var hOldPen As Long
    Var i As Long
    Var txt As String
    Var Ret As Long

    SetFontName "MS 明朝"
    txt = "札幌市白石区"
    lf.lfHeight = 60
    lf.lfWeight = 400
    lf.lfCharSet = 128

    rFont = Api_CreateFontIndirect(lf)

    hDC = Api_GetDC(GethWnd)
    Ret = Api_SetPolyFillMode(hDC, WINDING)

    'パス用ペンを設定
    hPen = Api_CreatePen(PS_SOLID, 1, vbBlack)
    hOldPen = Api_SelectObject(hDC, rFont)
    Ret = Api_SetBkMode(hDC, TRANSPARENT)

    '通常の文字列描画
    Ret = Api_TextOut(hDC, 10, 5, txt, Len(txt))

    'パスブラケットをオープン
    Ret = Api_BeginPath(hDC)
    Ret = Api_TextOut(hDC, 10, 70, txt, Len(txt))
    Ret = Api_EndPath(hDC)

    Ret = Api_WidenPath(hDC)

    'クリッピング領域とパスを結合
    Ret = Api_SelectClipPath(hDC, RGN_COPY)

    'オブジェクトを選択
    Ret = Api_SelectObject(hDC, hOldPen)

    'パス用ペンを設定
    hPen = Api_CreatePen(PS_SOLID, 1, vbBlack)
    hOldPen = Api_SelectObject(hDC, hPen)

    For i = 71 To 129
        Ret = Api_MoveToEx(hDC, 11, i, 0)
        Ret = Api_LineTo(hDC, 370, i)
    Next

    '論理オブジェクトを削除
    Ret = Api_DeleteObject(hPen)

    'デバイスコンテキストを解放
    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

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

クリッピング領域の設定

クリッピング領域を新たに設定します。
SetRect RECT構造体の値を設定
IntersectClipRect 指定したデバイスコンテキストのクリッピング領域と、指定した長方形が重なる部分を新しいクリッピング領域に設定
DrawEdge 矩形に3D効果を与える
SelectClipRgn クリッピング領域を設定
FillRect ブラシで矩形領域を塗り潰す
CreateSolidBrush ソリッドカラーで論理ブラシを作成
Ellippse 楕円の描画
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

例では、「SetRect」をクリックした場合は、指定した領域に指定したサイズで描画。
「実行」をクリックした場合は、IntersectClipRectで指定された領域にクリップされた状態を表しています。


```
'=====================================
'= クリッピング領域の設定
'=    (IntersectClipRect.bas)
'=====================================
#include "Windows.bi"

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

#define BF_BOTTOM &H8                       '下辺を描画
#define BF_LEFT &H1                         '左辺を描画
#define BF_RECT (&H1 Or &H2 Or &H4 Or &H8)  '(BF_LEFT Or BF_TOP Or BF_RIGHT Or BF_BOTTOM)
#define BF_RIGHT &H4                        '右辺を描画
#define BF_TOP &H2                          '上辺を描画
#define BF_MIDDLE &H800                     ' Fill in the middle.
#define BDR_INNER &HC                       '
#define BDR_OUTER &H3                       '
#define BDR_RAISED &H5                      '
#define BDR_RAISEDINNER &H4                 '内側が凸
#define BDR_RAISEDOUTER &H1                 '外側エッジが凸
#define BDR_SUNKEN &HA                      '
#define BDR_SUNKENINNER &H8                 '内側が凹
#define BDR_SUNKENOUTER &H2                 '外側エッジが凹
#define EDGE_RAISED (&H1 Or &H4)            '(BDR_RAISEDOUTER Or BDR_RAISEDINNER)

' 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&)

' 指定したデバイスコンテキストのクリッピング領域と、指定した長方形が重なる部分を、新しいクリッピング領域に設定
Declare Function Api_IntersectClipRect& Lib "gdi32" Alias "IntersectClipRect" (ByVal
hDC&, ByVal X1&, ByVal Y1&, ByVal X2&, ByVal Y2&)

' 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&)

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

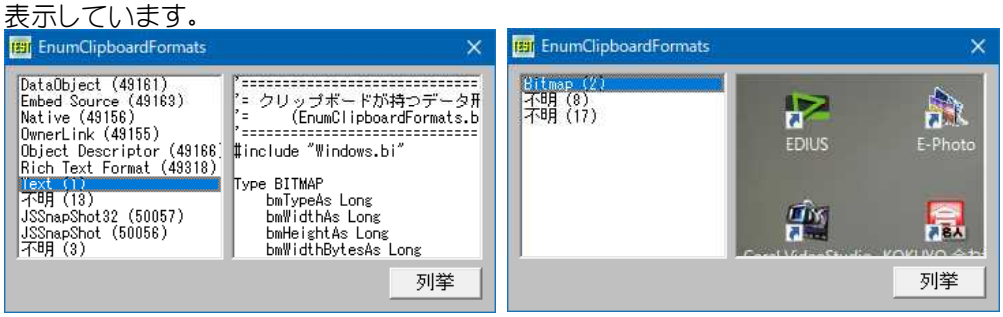
```
' ブラシで矩形領域を塗りつぶす
Declare Function Api_FillRect& Lib "user32" Alias "FillRect" (ByVal hDC&, ByRef r As RECT,
ByVal hBrush&)

' 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&)

' 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&)

'=====================================
'= 新しいクリッピング領域
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var rct1 As RECT
    Var rct2 As RECT
    Var hDC As Long
    Var Ret As Long

    hDC = Api_GetDC(GethWnd)

    Cls

    Ret = Api_IntersectClipRect(hDC, 20, 10, 130, 95)

    rct1.Right = GetWidth
    rct1.Bottom = GetHeight
    Ret = Api_FillRect(hDC, rct1, Api_CreateSolidBrush(255))

    Ret = Api_Ellipse(hDC, 50, 60, 200, 190)

    rct2.Left = 70
    rct2.Top = 5
    rct2.Right = 200
    rct2.Bottom = 50
    Ret = Api_DrawEdge(hDC, rct2, EDGE_RAISED, BF_RECT Or BF_MIDDLE)

    Ret = Api_SelectClipRgn(hDC, 0)
    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

'=====================================
'= 通常領域
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var rct1 As RECT
    Var rct2 As RECT
    Var hDC As Long
    Var Ret As Long

    hDC = Api_GetDC(GethWnd)

    Cls

    rct1.Right = GetWidth
    rct1.Bottom = GetHeight
    Ret = Api_SetRect(rct1, 20, 10, 130, 95)
    Ret = Api_FillRect(hDC, rct1, Api_CreateSolidBrush(255))

    Ret = Api_Ellipse(hDC, 50, 60, 200, 190)
```

```
    rct2.Left = 70
    rct2.Top = 5
    rct2.Right = 200
    rct2.Bottom = 50
    Ret = Api_DrawEdge(hDC, rct2, EDGE_RAISED, BF_RECT Or BF_MIDDLE)

    Ret = Api_SelectClipRgn(hDC, 0)
    Ret = Api_ReleaseDC(GethWnd, hDC)
End Sub

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

クリップボードが持つデータ形式の列挙と表示

OpenClipboard クリップボードをオープン
CloseClipboard クリップボードをクローズ
EnumClipboardFormats クリップボード内にあるフォーマットを列挙
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
GetClipboardFormatName クリップボードのフォーマットから名前を取得
lstrlen 指定された文字列のバイトまたは文字の長さを返す
lstrcpy 文字列をコピーする
GlobalLock ヒープに確保されたメモリをロック
GlobalUnlock メモリブロックのロックを解除
GetObject オブジェクト取得
BitBlt ビットブロック転送
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
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
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、最初文字列を、次にデスクトップをコピーした場合のデータ形式の列挙と、形式項目を選択した場合のデータを表示しています。




```
'=======================================
'= クリップボードが持つデータ形式の列挙と表示
'=     (EnumClipboardFormats.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
```

```
' 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" ()

' クリップボード内にあるフォーマットを列挙
Declare Function Api_EnumClipboardFormats& Lib "user32" Alias "EnumClipboardFormats"
(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&)

' クリップボードのフォーマットから名前を取得
Declare Function Api_GetClipboardFormatName& Lib "user32" Alias
"GetClipboardFormatNameA" (ByVal wFormat&, ByVal lpString$, ByVal nMaxCount&)

' 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$)

' 文字列をコピーする
Declare Function Api_lstrcpy& Lib "Kernel32" Alias "lstrcpyW" (lpszString1 As Any,
lpszString2 As Any)

' 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&)

' 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&)

' 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 CF_BITMAP 2                         'ビットマップのデータ(HBITMAP)
#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

Var Shared List1 As Object
Var Shared Picture1 As Object
Var Shared Button1 As Object

List1.Attach GetDlgItem("List1") : List1.SetFontSize 12
List1.SetWindowSize 152, 136
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14
Picture1.Attach GetDlgItem("Picture1")

Var Shared mhDC As Long
Var Shared phDC As Long
```

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

    'ピクチャボックスのデバイスコンテキストを取得
    phDC = Api_GetDC(Picture1.GethWnd)

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

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var lFormat As Long
    Var sFormat As String * 256
    Var i As Integer
    Var Ret As Long

    List1.ResetContent
    Picture1.Cls

    Ret = Api_OpenClipboard(GethWnd)

    Do
        i = i + 1
        lFormat = Api_EnumClipboardFormats(lFormat)
        If lFormat = 0 Then Exit Do

        Select Case lFormat
            Case CF_TEXT
                sFormat = "Text" & Chr$(0)
            Case CF_BITMAP
                sFormat = "Bitmap" & Chr$(0)
            Case CF_METAFILEPICT
                sFormat = "CF_METAFILEPICT" & Chr$(0)
            Case CF_SYLK
                sFormat = "CF_SYLK" & Chr$(0)
            Case CF_DIF
                sFormat = "CF_DIF" & Chr$(0)
            Case CF_TIFF
                sFormat = "CF_TIFF" & Chr$(0)
            Case CF_OEMTEXT
                sFormat = "CF_OEMTEXT" & Chr$(0)
            Case CF_DIB
                sFormat = "CF_DIB" & Chr$(0)
            Case CF_PALETTE
                sFormat = "CF_PALETTE" & Chr$(0)
            Case CF_PENDATA
                sFormat = "CF_PENDATA" & Chr$(0)
            Case CF_RIFF
                sFormat = "CF_RIFF" & Chr$(0)
            Case CF_WAVE
                sFormat = "CF_WAVE" & Chr$(0)
            Case CF_UNICODETEXT
                sFormat = "CF_UNICODETEXT" & Chr$(0)
            Case CF_ENHMETAFILE
                sFormat = "CF_ENHMETAFILE" & Chr$(0)
            Case CF_HDROP
                sFormat = "CF_HDROP" & Chr$(0)
            Case CF_LOCALE
                sFormat = "CF_LOCALE" & Chr$(0)
            Case CF_DIBV5
                sFormat = "CF_DIBV5" & Chr$(0)
            Case Else
                If Api_GetClipboardFormatName(lFormat, sFormat, Len(sFormat)) = 0 Then
```

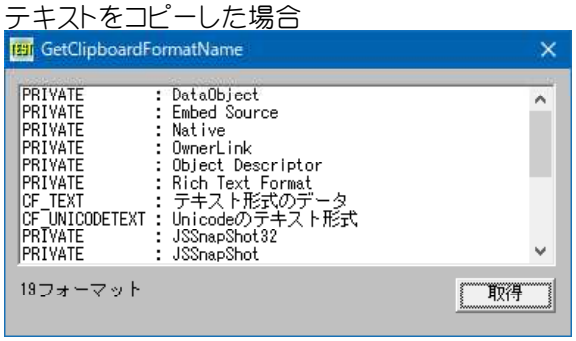
```
                sFormat = "不明" & Chr$(0)
            End If
        End Select

        List1.AddString Left$(sFormat, InStr(1, sFormat, Chr$(0)) - 1) & " (" & Trim$(Str$
(lFormat)) & ")"
    Loop

    Ret = Api_CloseClipboard
End Sub

'=====================================
'=
'=====================================
Declare Sub List1_Click edecl ()
Sub List1_Click()
    Var bmp As BITMAP
    Var hBit As Long
    Var hBitmap As Long
    Var TmpStr As String
    Var TmpStr2 As String
    Var StrLen As Long
    Var hTmpStr As Long
    Var pTmpStr As Long
    Var Ret As Long

    TmpStr2 = List1.GetText(List1.GetCursel)

    Select Case Left$(TmpStr2, InStr(TmpStr2, "(") - 2)
        Case "Text"
            Ret = Api_OpenClipboard(GethWnd)
            hTmpStr = Api_GetClipboardData(CF_TEXT)
            pTmpStr = Api_GlobalLock(hTmpStr)

            TmpStr = Space$(Api_lstrLen(ByVal pTmpStr))
            Ret = Api_lstrcpy(TmpStr, ByVal pTmpStr)
            Ret = Api_GlobalUnlock(hTmpStr)

            Picture1.Cls
            Picture1.SetFontName "MS ゴシック"
            Picture1.SetFontSize 12
            Picture1.Print TmpStr

        Case "Bitmap"
            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)

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

            'Object選択
            Ret = Api_SelectObject(mhDC, hBit)

            Picture1.Cls
            Picture1.SetFontSize 12

            '指定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)

        Case Else
            Picture1.Cls
            Picture1.SetFontSize 12
            Picture1.Print "表示できません！"
    End Select

    Ret = Api_CloseClipboard
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
```

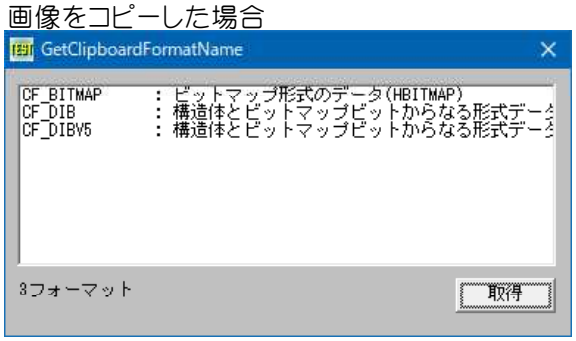
クリップボード内にあるフォーマットを列挙(Ⅰ)

クリップボード内にあるフォーマットを列挙します。
CountClipboardFormats クリップボード内のフォーマット数を取得
EnumClipboardFormats クリップボード内にあるフォーマットを列挙
OpenClipboard クリップボードをオープン
CloseClipboard クリップボードをクローズ
GetClipboardFormatName クリップボードのフォーマットから名前を取得

テキストをコピーした場合



画像をコピーした場合




```
'=====================================
'= クリップボード内にあるフォーマットを列挙(Ⅰ)
'=    (GetClipboardFormatName.bas)
'=====================================
#include "Windows.bi"

' クリップボード内のフォーマット数を取得
Declare Function Api_CountClipboardFormats& Lib "User32" Alias "CountClipboardFormats"
()

' クリップボード内にあるフォーマットを列挙
Declare Function Api_EnumClipboardFormats& Lib "User32" Alias "EnumClipboardFormats"
(ByVal wFormat&)

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

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

' クリップボードのフォーマットから名前を取得
Declare Function Api_GetClipboardFormatName& Lib "User32" Alias
"GetClipboardFormatNameA" (ByVal wFormat&, ByVal lpString$, ByVal nMaxCount&)

#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
```

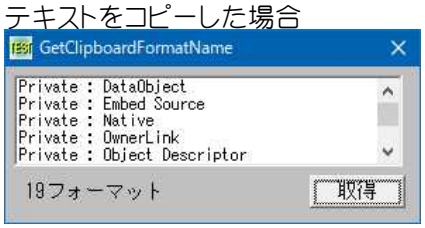
```
#define CF_BITMAP 2                         'ビットマップのデータ(HBITMAP)
#define CF_METAFILEPICT 3                   'メタファイル画像形式。METAFILEPICT構造体のメモリオブジェクト
#define CF_SYLK 4                           'Microsoftシンボリックリンク(SYLK)形式のデータ
#define CF_DIF 5                            'SoftwareArts社のDIFデータ交換形式
#define CF_TIFF 6                           'TIFF形式の画像データ
#define CF_OEMTEXT 7                        'OEM 文字セットの文字を持つテキスト形式データ
#define CF_DIB 8                            '構造体とビットマップビットからなるメモリオブジェクト
#define CF_PALETTE 9                        'カラーパレットのハンドル
#define CF_PENDATA 10                       'Windowsのペン拡張機能のためのデータ
#define CF_RIFF 11                          'RIFF 形式の音声データ
#define CF_WAVE 12                          'WAVE 形式の音声データ
#define CF_UNICODETEXT 13                   'Unicodeのテキスト形式
#define CF_ENHMETAFILE 14                   '拡張メタファイルのデータです(HENHMETAFILE)
#define CF_HDROP 15                         'HDROP型
#define CF_LOCALE 16                        'テキストデータのロケールIDハンドル
#define CF_DIBV5 17                         '構造体とビットマップビットからなるメモリオブジェクト(Windows2000)

Var Shared List1 As Object
Var Shared Text1 As Object
Var Shared Button1 As Object

List1.Attach GetDlgItem("List1") : List1.SetFontSize 12
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 12
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 12

'=====================================
'=
'=====================================
Declare Function FormatName(lFormatId As Long ) As String
Function FormatName(lFormatId As Long ) As String
    Var lSize As Long
    Var sBuf As String
    Var Ret As Long

    If (lFormatId >= 1 And lFormatId <= 17) Then
        Select Case lFormatId
            Case CF_TEXT
                FormatName = "CF_TEXT       : " & "テキスト形式のデータ"
            Case CF_BITMAP
                FormatName = "CF_BITMAP     : " & "ビットマップ形式のデータ(HBITMAP)"
            Case CF_METAFILEPICT
                FormatName = "CF_METAFILEPICT: " & "メタファイル画像形式"
            Case CF_SYLK
                FormatName = "CF_SYLK       : " & "シンボリックリンク(SYLK)形式のデータ"
            Case CF_DIF
                FormatName = "CF_DIF        : " & "SoftwareArts社のDIFデータ交換形式"
            Case CF_TIFF
                FormatName = "CF_TIFF       : " & "TIFF形式の画像データ"
            Case CF_OEMTEXT
                FormatName = "CF_OEMTEXT    : " & "OEM文字セットの文字を持つテキスト形式データ"
            Case CF_DIB
                FormatName = "CF_DIB        : " & "構造体とビットマップビットからなる形式データ"
            Case CF_PALETTE
                FormatName = "CF_PALETTE    : " & "カラーパレットのハンドル"
            Case CF_PENDATA
                FormatName = "CF_PENDATA    : " & "Windowsのペン拡張機能のためのデータ"
            Case CF_RIFF
                FormatName = "CF_RIFF       : " & "RIFF形式の音声データ"
            Case CF_WAVE
                FormatName = "CF_WAVE       : " & "WAVE形式の音声データ"
            Case CF_UNICODETEXT
                FormatName = "CF_UNICODETEXT : " & "Unicodeのテキスト形式"
            Case CF_ENHMETAFILE
                FormatName = "CF_ENHMETAFILE : " & "拡張メタファイルのデータ"
            Case CF_HDROP
                FormatName = "CF_HDROP      : " & "ファイルリスト(HDROP型)"
            Case CF_LOCALE
                FormatName = "CF_LOCALE     : " & "テキストデータのロケールIDハンドル"
            Case CF_DIBV5
```

```
                FormatName = "CF_DIBV5       : " & "構造体とビットマップビットからなる形式データ
(Windows2000)"
        End Select
    Else
        lSize = 255
        sBuf = String$(lSize, 0)
        Ret = Api_GetClipboardFormatName(lFormatId, sBuf, lSize)

        If Ret <> 0 Then
            FormatName = "PRIVATE       : " & Left$(sBuf, Ret)
        Else
            FormatName = "Unknown       : " & Left$(sBuf, Ret)
        End If
    End If
End Function

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

    List1.Resetcontent

    If (Api_OpenClipboard(GethWnd)) Then
        Ret = Api_EnumClipboardFormats(0)

        If (Ret <> 0) Then
            Do
                List1.AddString FormatName(Ret)
                Ret = Api_EnumClipboardFormats(Ret)
            Loop While Ret <> 0
        End If
    End If

    Ret = Api_CloseClipboard

    Text1.SetWindowText Trim$(Str$(Api_CountClipboardFormats)) & "フォーマット"
End Sub

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

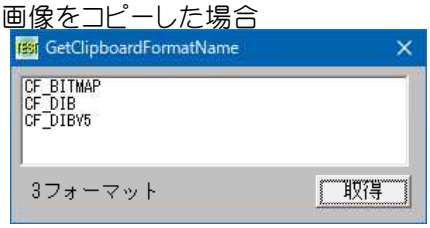
クリップボード内にあるフォーマットを列挙(II)

クリップボード内にあるフォーマットを列挙します。
EnumClipboardFormats クリップボード内にあるフォーマットを列挙
OpenClipboard クリップボードをオープン
CloseClipboard クリップボードをクローズ
GetClipboardFormatName クリップボードのフォーマットから名前を取得

テキストをコピーした場合



画像をコピーした場合



```
'====================================
'= クリップボード内のフォーマットを列挙(II)
'=    (GetClipboardFormatName2.bas)
'====================================
#include "Windows.bi"

#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 CF_METAFILEPICT 3                  'メタファイル画像形式。METAFILEPICT構造体のメモリオブ
                                           ジェクト
#define CF_SYLK 4                          'Microsoftシンボリックリンク(SYLK)形式のデータ
#define CF_DIF 5                           'SoftwareArts社のDIFデータ交換形式
#define CF_TIFF 6                          'TIFF形式の画像データ
#define CF_OEMTEXT 7                       'OEM文字セットの文字を持つテキスト形式データ
#define CF_DIB 8                           '構造体とビットマップビットからなるメモリオブジェクト
#define CF_PALETTE 9                       'カラーパレットのハンドル
#define CF_PENDATA 10                      'Windowsのペン拡張機能のためのデータ
#define CF_RIFF 11                         'RIFF形式の音声データ
#define CF_WAVE 12                         'WAVE形式の音声データ
#define CF_UNICODETEXT 13                  'Unicodeのテキスト形式
#define CF_ENHMETAFILE 14                  '拡張メタファイルのデータです(HENHMETAFILE)
#define CF_FILES 15                        'FILES
#define CF_HDROP 15                        'HDROP型
#define CF_LOCALE 16                       'テキストデータのロケールIDハンドル
#define CF_DIBV5 17                        '
#define CF_OWNERDISPLAY &H80               'オーナー表示形式
#define CF_DSPTEXT &H81                    'プライベートな形式のテキストデータ
#define CF_DSPBITMAP &H82                  'プライベートな形式のビットマップデータ
#define CF_DSPMETAFILEPICT &H83            'プライベートな形式の拡張メタファイルデータ
#define CF_DSPENHMETAFILE &H8E             'プライベートな形式の拡張メタファイルデータ

#define ERROR_SUCCESS &H0                  '正常終了の戻り値を示す
#define LB_SETTABSTOPS &H192               'リストボックス内のタブストップを設定する

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

' クリップボード内にあるフォーマットを列挙
Declare Function Api_EnumClipboardFormats& Lib "user32" Alias "EnumClipboardFormats"
(ByVal wFormat&)

' クリップボードのフォーマットから名前を取得
Declare Function Api_GetClipboardFormatName& Lib "user32" Alias
"GetClipboardFormatNameA" (ByVal wFormat&, ByVal lpString$, ByVal nMaxCount&)

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

Var Shared List1 As Object
Var Shared Text1 As Object
Var Shared Button1 As Object

List1.Attach GetDlgItem("List1") : List1.SetFontSize 12
Text1.Attach GetDlgItem("Text1") : Text1.SetFontSize 14
Button1.Attach GetDlgItem("Button1") : Button1.SetFontSize 14

'====================================
'=
'====================================
Declare Function GetCBTypeDesc(ByVal nFormat As Long) As String
Function GetCBTypeDesc(ByVal nFormat As Long) As String
    Var cbBuff As Long
    Var Buff As String

    Select Case nFormat
       Case CF_TEXT
          Buff = "CF_TEXT"
       Case CF_BITMAP
          Buff = "CF_BITMAP"
```

```
      Case CF_METAFILEPICT
         Buff = "CF_METAFILEPICT"
      Case CF_SYLK
         Buff = "CF_SYLK"
      Case CF_DIF
         Buff = "CF_DIF"
      Case CF_TIFF
         Buff = "CF_TIFF"
      Case CF_OEMTEXT
         Buff = "CF_OEMTEXT"
      Case CF_DIB
         Buff = "CF_DIB"
      Case CF_PALETTE
         Buff = "CF_PALETTE"
      Case CF_PENDATA
         Buff = "CF_PENDATA"
      Case CF_RIFF
         Buff = "CF_RIFF"
      Case CF_WAVE
         Buff = "CF_WAVE"
      Case CF_UNICODETEXT
         Buff = "CF_UNICODETEXT"
      Case CF_ENHMETAFILE
         Buff = "CF_ENHMETAFILE"
      Case CF_FILES, CF_HDROP
         Buff = "CF_FILES/CF_HDROP"
      Case CF_LOCALE
         Buff = "CF_LOCALE"
      Case CF_DIBV5
         Buff = "CF_DIBV5"
      Case CF_OWNERDISPLAY
         Buff = "CF_OWNERDISPLAY"
      Case CF_DSPTEXT
         Buff = "CF_DSPTEXT"
      Case CF_DSPBITMAP
         Buff = "CF_DSPBITMAP"
      Case CF_DSPMETAFILEPICT
         Buff = "CF_DSPMETAFILEPICT"
      Case CF_DSPENHMETAFILE
         Buff = "CF_DSPENHMETAFILE"
      Case Else
         Buff = String$(256, Chr$(0))
         cbBuff = Len(Buff)

         If Api_GetClipboardFormatName(nFormat, Buff, cbBuff) = 0 Then
            Buff = "Unknown : " & Str$(nFormat)
         Else
            Buff = "Private : " & Buff
         End If
   End Select

   GetCBTypeDesc = Buff
End Function

'=====================================
'=
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
   Var fmt As Long

   List1.ResetContent
   If Api_OpenClipboard(0) <> 0 Then
      Do
         fmt = Api_EnumClipboardFormats(fmt)

         If fmt > ERROR_SUCCESS Then
            List1.AddString GetCBTypeDesc(fmt)
         End If
      Loop Until fmt = ERROR_SUCCESS
```

```
        Ret = Api_CloseClipboard
    End If

    Text1.SetWindowText Str$(List1.GetCount) & "フォーマット"
End Sub

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

クリップボードのシーケンス番号を取得

OpenClipboard クリップボードをオープン
CloseClipboard クリップボードをクローズ
EmptyClipboard クリップボードを空にする
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
GetClipboardSequenceNumber 現在のウィンドウステーションのクリップボードのシーケンス番号を取得
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
GlobalLock ヒープに確保されたメモリをロック
GlobalUnlock メモリブロックのロックを解除
GlobalSize グローバルメモリオブジェクトのサイズを取得
MoveMemory メモリの指定領域をコピー
lstrcpy 文字列をコピーする


```
'=====================================
'= クリップボードのシーケンス番号を取得
'=    (GetClipboardSequenceNumber.bas)
'=====================================
#include "Windows.bi"

' 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&)

' 現在のウィンドウステーションのクリップボードのシーケンス番号を取得
Declare Function Api_GetClipboardSequenceNumber& Lib "user32" Alias
"GetClipboardSequenceNumber" ()
```

```
' 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&)
' 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&)

' グローバルメモリオブジェクトのサイズを取得
Declare Function Api_GlobalSize& Lib "kernel32" Alias "GlobalSize" (byval hMem&)

' メモリの指定領域をコピー
Declare Sub MoveMemory Lib "kernel32" Alias "RtlMoveMemory" (ByVal Destination&, ByVal
Source&, ByVal Length&)

' 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 GMEM_DDESHARE &H2000              'DDE 関数が使用するメモリを確保する
#define GMEM_FIXED &H0                    '固定メモリブロックを割り当てる
#define GMEM_MOVEABLE &H2                 '移動可能メモリブロックを割り当てる
#define GMEM_ZEROINIT &H40                '割り当てたメモリブロックをゼロで初期化する
#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 Text(2) As Object
Var Shared Button(1) As Object

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
    If i < 2 Then
        Button(i).Attach GetDlgItem("Button" & Trim$(Str$(i + 1))) :
Button(i).SetFontSize 14
    End If
Next

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

    MyStr = Edit1.GetWindowText

    If Api_OpenClipboard(ByVal 0) Then

        'クリップボードを空にする
        Ret = Api_EmptyClipboard

        'Nullの分を追加(+1)
        Size = Len(MyStr) + 1
        hMem = Api_GlobalAlloc(GMEM_MOVEABLE, Size)

        'メモリをロック
        lpMem = Api_GlobalLock(hMem)
        MoveMemory lpMem, StrAdr(MyStr), Size
```

```
        'ロックを解除
        Ret = Api_GlobalUnlock(hMem)

        'クリップボードにコピー
        Ret = Api_SetClipboardData(CF_TEXT, hMem)

        'メモリブロックのロックを解放
        Ret = Api_GlobalFree(hMem)

        'クリップボードを閉じる
        Ret = Api_CloseClipboard()
    End If
End Sub

'=====================================
'= クリップボードデータ(TEXT)を取り出す
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var hMem As Long
    Var lpMem As Long
    Var MyStr As String
    Var Ret As Long

    If Api_OpenClipboard(0) = 0 Then
        A% = MessageBox("", "クリップボードが開きません", 0, 0)
        Exit Sub
    End If

    Text(0).SetWindowText ""

    '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
        MyStr = Space$(MAXSIZE)
        Ret = Api_lstrcpy(MyStr, lpMem)
        Ret = Api_GlobalUnlock(hMem)

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

    Text(1).SetWindowText Str$(Api_GetClipboardSequenceNumber)

    Ret = Api_CloseClipboard()
End Sub

'=====================================
'=
'=====================================
Declare Sub Edit1_SetFocus edecl ()
Sub Edit1_SetFocus()
    Text(0).SetWindowText ""
End Sub

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

クリップボードビューアチェイン内のハンドルを取得

GetClipboardViewer クリップボードビューアチェイン内の最初のウィンドウハンドルを取得

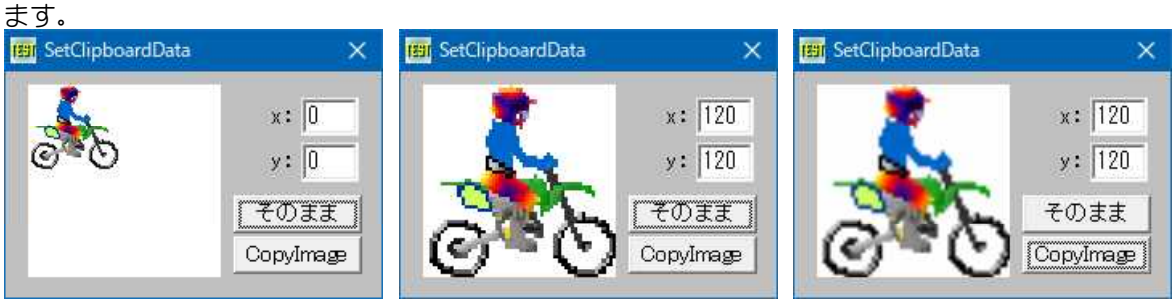
クリップボードビューアのハンドルを取得し、結果をMiniMiniSpySで確認しています。


```
'=======================================
'= クリップボードビューアチェイン内のハンドルを取得
'=     (GetClipboardViewer.bas)
'=======================================
#include "Windows.bi"

' クリップボードビューアチェイン内の最初のウィンドウハンドルを取得
Declare Function Api_GetClipboardViewer& Lib "user32" Alias "GetClipboardViewer" ()

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 ViewerHandle As Long

    'クリップボードビューアチェインのハンドル取得
    ViewerHandle = Api_GetClipboardViewer

    If ViewerHandle = 0 Then                    '取得結果がNULLのときは
        Text1.SetWindowtext "チェインは存在しません。"

    Else                                        '取得結果がNULL以外のときは
        Text1.SetWindowText "Viewer Handle : &&H" & Hex$(ViewerHandle)
    End If
End Sub

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

クリップボードへ転送と取り出し(BITMAP)

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

例では、既存のBMPファイル「bike1.bmp:56x52Dot」を読み込んでいます。
「そのまま」:LoadImageでnx、nyのサイズで読み込んだBITMAPをSetClipboardData(CF_BITMAP)でクリップボードに設定(転送)しています。
　nx、nyが「0」の場合は元のサイズで読み込みます。
「CopyImage」:LoadImageで元のサイズで読み込みます。CopyImageでnx、nyのサイズにコピー後、SetClipboardData(CF_BITMAP)でクリップボードに設定(転送)しています。
どちらの場合もクリップボードに転送されたデータを即、そのサイズでPictureBox(120x120Dot)に描画させています。




```
'=======================================
'= BITMAPのクリップボード転送と取り出し
'=    (SetClipboardData.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_DEFAULTSIZE &H40             '標準サイズで表示
#define LR_COPYRETURNORG &H4            '指定されたサイズを無視し、そのままコピー
#define LR_SHARED &H8000                'イメージハンドルを固定する
#define LR_LOADFROMFILE &H10            '外部ファイルからロードする

#define CF_BITMAP 2                     'ビットマップのデータ(HBITMAP)
#define CF_DIB 8                        '構造体とビットマップビットからなるメモリオブジェクト
#define CF_DIF 5                        'SoftwareArts社のDIFデータ交換形式
#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_TIFF 6                       'TIFF形式の画像データ
#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&)

' イメージを拡大縮小してコピーする
Declare Function Api_CopyImage& Lib "user32" Alias "CopyImage" (ByVal handle&, ByVal
imageType&, ByVal newWidth&, ByVal newHeight&, ByVal lFlags&)

' 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&)

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

' 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&)

' 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 Picture1 As Object
Var Shared Text(1) As Object
Var Shared Edit(1) As Object
Var Shared Button(1) As Object

Picture1.Attach GetDlgItem("Picture1")
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
    Button(i).Attach GetDlgItem("Button" & Trim$(Str$(i + 1))) : Button(i).SetFontSize 14
Next

'=====================================
'= ClipBoardからPictureBoxへ
'=====================================
Declare Sub ImageFromClipBoard ()
```

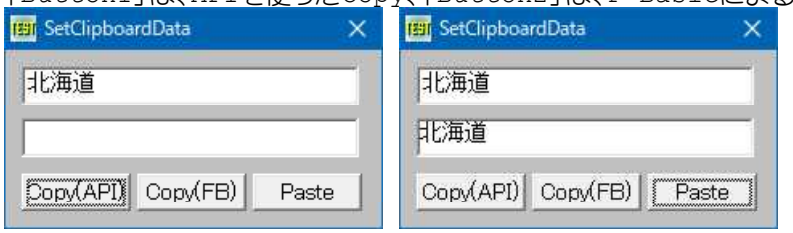
```
Sub ImageFromClipBoard()
    Var bmp As BITMAP
    Var hBit As Long
    Var phDC As Long
    Var mhDC As Long
    Var Ret As Long

    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

'=====================================
'= LoadImage → SetClipboardData
'=====================================
Declare Sub Button1_on edecl ()
Sub Button1_on()
    Var hBit As Long
    Var nx As Long
    Var ny As Long
    Var Ret As Long

    Ret = Api_OpenClipboard(GethWnd)
    Ret = Api_EmptyClipboard

    nx = Val(Edit(0).GetWindowText)
    ny = Val(Edit(1).GetWindowText)

    '画像ファイル(bike1.bmp)を読み込む
    hBit = Api_LoadImage(0, "bike1.bmp", IMAGE_BITMAP, nx, ny, LR_LOADFROMFILE)

    'クリップボードにデータを設定
    Ret = Api_SetClipboardData(CF_BITMAP, hBit)

    Ret = Api_CloseClipboard

    ImageFromClipBoard
End Sub

'=====================================
'= LoadImage → CopyImage → SetClipboardData
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var hBit As Long
    Var hNew As Long
```

```
    Var nx As Long
    Var ny As Long
    Var Ret As Long

    Ret = Api_OpenClipboard(GethWnd)
    Ret = Api_EmptyClipboard

    nx = Val(Edit(0).GetWindowText)
    ny = Val(Edit(1).GetWindowText)

    '画像ファイル(bike1.bmp)を読み込む
    hBit = Api_LoadImage(0, "bike1.bmp", IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE)

    'イメージを拡大縮小してコピー
    hNew = Api_CopyImage(hBit, IMAGE_BITMAP, nx, ny, LR_COPYRETURNORG)

    'クリップボードにデータを設定
    Ret = Api_SetClipboardData(CF_BITMAP, hNew)
    Ret = Api_CloseClipboard

    ImageFromClipBoard
End Sub

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

クリップボードへ転送と取り出し(TEXT)

クリップボートへテキストを転送(Copy)し、それを取り出し(Paste)ます。
OpenClipboard クリップボードをオープン
CloseClipboard クリップボードをクローズ
EmptyClipboard クリップボードを空にする
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
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
GlobalLock ヒープに確保されたメモリをロック
GlobalUnlock メモリブロックのロックを解除
GlobalSize グローバルメモリオブジェクトのサイズを取得
MoveMemory メモリの指定領域をコピー
lstrcpy 文字列をコピーする

「Button1」は、APIを使ったCopy、「Button2」は、F-BasicによるCopyです。




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

' 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&)

' 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&)

' 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&)

' グローバルメモリオブジェクトのサイズを取得
Declare Function Api_GlobalSize& Lib "kernel32" Alias "GlobalSize" (byval hMem&)

' メモリの指定領域をコピー
Declare Sub MoveMemory Lib "kernel32" Alias "RtlMoveMemory" (ByVal Destination&, ByVal
Source&, ByVal Length&)

' 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 GMEM_DDESHARE &H2000              'DDE 関数が使用するメモリを確保する
#define GMEM_FIXED &H0                    '固定メモリブロックを割り当てる
#define GMEM_MOVEABLE &H2                 '移動可能メモリブロックを割り当てる
#define GMEM_ZEROINIT &H40                '割り当てたメモリブロックをゼロで初期化する
#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 Button(2) As Object

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

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

    MyStr = Edit1.GetWindowText

    If Api_OpenClipboard(ByVal 0) Then
```

```
        'クリップボードを空にする
        Ret = Api_EmptyClipboard

        'Nullの分を追加(+1)
        Size = Len(MyStr) + 1
        hMem = Api_GlobalAlloc(GMEM_MOVEABLE, Size)

        'メモリをロック
        lpMem = Api_GlobalLock(hMem)
        MoveMemory lpMem, StrAdr(MyStr), Size

        'ロックを解除
        Ret = Api_GlobalUnlock(hMem)

        'クリップボードにコピー
        Ret = Api_SetClipboardData(CF_TEXT, hMem)

        'メモリブロックのロックを解放
        Ret = Api_GlobalFree(hMem)

        'クリップボードを閉じる
        Ret = Api_CloseClipboard()
    End If
End Sub

'=====================================
'= F-Basicでクリップボードへコピー
'=====================================
Declare Sub Button2_on edecl ()
Sub Button2_on()
    Var MyStr As String

    SetMapMode 0
    ClearCB
    SetCBFormat "CF_TEXT"

    Edit1.SetSelText 0, -1
    Edit1.Copy
    Edit1.SetSelText -1, -1
End Sub

'=====================================
'= クリップボードデータ(TEXT)を取り出す
'=====================================
Declare Sub Button3_on edecl ()
Sub Button3_on()
    Var hMem As Long
    Var lpMem As Long
    Var MyStr As String
    Var Ret As Long

    If Api_OpenClipboard(0) = 0 Then
        A% = MessageBox("", "クリップボードが開きません", 0, 0)
        Exit Sub
    End If

    Text1.SetWindowText ""

    '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
        MyStr = Space$(MAXSIZE)
        Ret = Api_lstrcpy(MyStr, lpMem)
        Ret = Api_GlobalUnlock(hMem)
```

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

    Ret = Api_CloseClipboard()
End Sub

'=======================================
'=
'=======================================
Declare Sub Edit1_SetFocus edecl ()
Sub Edit1_SetFocus()
    Text1.SetWindowText ""
End Sub

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