

NEW TECHNOLOGY KIT FOR [X]HARBOUR - AVAILABLE CORE FUNCTIONS

FUNCTION NAME BY CATEGORY	ADDED	FINISHED	UPDATED
ACCELERATORS			
NTK_CopyAcceleratorTable(hAcceSrc, aAcceDest) -> nEntries	< 12.31.02	YES	
NTK_CreateAcceleratorTable(aAcce) -> hAcce	< 12.31.02	YES	
NTK_DestroyAcceleratorTable(hAcceSrc) -> lRet	< 12.31.02	YES	
NTK_LoadAccelerators(hInstance, cAcceName) -> hAcce	< 12.31.02	YES	
NTK_SetAccelerator(aAcce hResAcce) -> hAcce	< 12.31.02	YES	
BITS			
NTK_And()	08.12.03	YES	
NTK_CheckBit()	10.31.03	YES	
NTK_HiByte()	08.26.06	YES	
NTK_HiWord()	12.31.02	YES	
NTK_LoWord()	< 12.31.02	YES	
NTK_LoByte()	08.26.06	YES	
NTK_LShift(nExpr, nShiftPos)	17.01.07	YES	
NTK_MakeLParam(nLow, nHigh) -> LParam	10.31.03	YES	
NTK_MakeLong()	10.31.03	YES	
NTK_MakePoint(nX, nY) -> (POINT) pt	10.31.03	YES	// Make a C POINT structure
NTK_MakeWParam()	10.31.03	YES	
NTK_Not()	08.12.03	YES	
NTK_Or()	08.12.03	YES	
NTK_RShift(nExpr, nShiftPos)	17.01.07	YES	
NTK_SetBit()	10.31.03	YES	
NTK_Xor()	08.12.03	YES	
BUTTONS & TOOLBARS			
NTK_BtnNew(aBtnList, nBId, nBY, nBX, nBHeight, nBWidth, cBCaption, nBType,; bExec, nBACckKey, cBHlpMsg, lBState, nBStyle, hBFont,; hBFClr, hBBClr, ncBmpUp, ncBmpDn, ncBmpGray, nBBmp, nBBmpX,; hBBkgBmpBrush, hwndP) -> aBtnList			// Create a new Ntk easy Button object // See NTKBTN.ch
NTK_BtnOwnerDrawn(aBtnList, hwnd, nMsg, nwParam, nlParam) -> 0 (always Zero)			// Auto-paint Ntk's OD style buttons. // Must be use in response/Return of a WM_DRAWITEM msg
NTK_OnBtn(aMsg, aBtnList, [cEvtActionMode], [cMaskEvtFlt]) -> LastBtnID			// wait state + action detector for Ntk easy Button object. // Should be use inside a getmessge loop...
NTK_EnableBtn(hBtnWnd, lEnable) -> lBtnState			// Enable or Disable a button using its handle
NTK_EnableBtnID(nBtnID, lEnable) -> nBtnState			// Enable or Disable a button using its ID
NTK_TBAddMember(aTbBtnList, BtID, BtTitle, BtAcckKey, btAction, ; BtrBmpUP, BtrBmpDN, BtrBmpOff, BtrBmpOvr, ; BtvShift, btHShift, BtHlpMsg) -> aTbBtnList			// Declares a new toolbar member (button) into an aTbBtnList array
NTK_TBCreate(hwnd, nDirection, nTop, nLeft, nBtnHeight, nBtnWidth, nBtnSpacing, ; nBtnType, nBtnStyle, hBtnFont, aTbBtnList) -> nCreatedBtn or -1			// Creates a complete toolbar of buttons from the previously defined aTbBtnList array.
NTK_ToggleBtnGetCheck(nBtnID) -> BST_CHECKED BST_UNCHECKED BST_UNDETERMINATE			// Retrieve the current status of a STD or OD Toggle Button
NTK_ToggleBtnSetCheck(nBtnID, nStatus) -> 0			// Change the current status of a STD or OD Toggle Button
CARET			
NTK_CreateCaret()	04.01.04	YES	
NTK_DestroyCaret()	04.01.04	YES	
NTK_GetCaretBlinkTime(nRate)	04.01.04	YES	
NTK_GetCaretPos(aPt) -> lRet (Check aPt)	10.28.03	YES	
NTK_GetCaretX() -> nxpos	10.28.03	YES	
NTK_GetCaretY() -> nypos	10.28.03	YES	
NTK_HideCaret()	04.01.04	YES	
NTK_Hourglass(lShift) -> holdCursor	04.01.04	YES	// Return the previous used cursor
NTK_SetCaret(nCaretShape, hwnd, hEditFont) -> nil	04.01.04	YES	
NTK_SetCaretBlinkTime()	04.01.04	YES	
NTK_SetCaretPos()	10.28.03	YES	
NTK_SetEditInsCaret(nCaretShape)	28.05.04	YES	
NTK_SetEditOvrCaret(nCaretShape)	28.05.04	YES	
NTK_ShowCaret()	04.01.04	YES	

CURSOR		
NTK_GetCursorPos(void) -> aXYpos() // {nPosX, nPosY}	10.29.03	YES
NTK_GetCursorX(void) -> nPosX	10.29.03	YES
NTK_GetCursorY(void) -> nPosY	10.29.03	YES
NTK_LoadCursor()	< 12.31.02	YES
NTK_SetCursor()	< 12.31.02	YES
NTK_SetCursorPos()	10.29.03	YES
NTK_ShowCursor(lShow) -> nDispCounter	27.07.05	YES
CLIPBOARD		
NTK_CloseClipboard() -> lRet	03.30.04	YES
NTK_CountClipbFormats() -> nCountFmt	03.30.04	YES
NTK_EmptyClipboard() -> lRet	03.30.04	YES
NTK_EnumClipbFormats(nFmt) -> nNextFmt	03.30.04	YES
NTK_GetClipbData(nFmt) -> cData	03.30.04	YES
NTK_GetClipbFormatName(nFmtName) -> cFmtName	03.30.04	YES
NTK_GetClipbOwner() -> hwnd	03.30.04	YES
NTK_GetClipbViewer() -> hwnd	03.30.04	YES
NTK_IsClipbFormatAvailable(nFmt) -> lRet	03.30.04	YES
NTK_OpenClipboard(hwndOwner) -> lRet	03.30.04	YES
NTK_RegClipbFormat()	03.30.04	YES
NTK_SetClipbData(nFmt, cData) -> hData	03.30.04	YES
NTK_PasteClipbText(hwnd) -> cTxtData	03.30.04	YES
NTK_PutClipbText(hwnd, cTxtData) -> hData	03.30.04	YES
COMMON DIALOGS		
NTK_ChooseColor(nInitColor, aColors, nFlags) -> nColor	09.01.03	YES
NTK_GetDirDlg(hwndP, cRootPath, cSubTitle, nFlags) -> cPath	12.14.05	YES // Pick a folder using windows explorer tree
NTK_GetOpenFileName(hwnd, cFile, cTitle, aFilter, nFlags, cPath, cDefExt, nFilterIndex) -> cFilename	08.31.03	YES
NTK_GetSaveFileName(hwnd, cFile, cTitle, aFilter, nFlags, cPath, cDefExt, nFilterIndex) -> cFilename	09.01.03	YES
NTK_OpenFile(cFileName, @cOFStruct, nStyle) -> hFileHandle	01.20.06	YES // Creates, opens, reopens, or deletes a file.
NTK_SHBrowseForFolder(hwndP, cPidlRoot, cTitle, nFlags, nIdxImage) -> cPIDL		YES // Pick a folder using windows explorer tree
DIALOGS		
NTK_AppendDialog()	06.23.03	YES
NTK_CreateDialog()	06.23.03	YES
NTK_CreateDialogIndirect()	06.23.03	YES
NTK_DialogBox()	08.26.03	YES
NTK_GetDialogBaseUnits	06.23.03	YES
NTK_GetDlgCtrlID()	07.11.03	YES
NTK_GetDlgItem()	07.11.03	YES
NTK_GetDlgItemText()	07.11.03	YES
NTK_DlgUnitToPixel()	06.25.03	YES
NTK_EndDialog()	06.23.03	YES
NTK_ModalDialog()	06.23.03	YES
NTK_PixelToDlgUnit()	06.25.03	YES
NTK_MsgBox()	< 12.31.02	YES
NTK_SetDlgItem(hwnd, nIDBtn, cnText) -> lRet	07.11.03	YES
NTK_SendDlgItemMessage(<hwndDlg>, <nID>, <nMsg>, <nwParam>, <nlParam>) -> nRet	07.11.03	YES
NTK_CheckDlgButton(hwnd, nIDBtn, nChkState) -> lRet	08.10.06	YES // Changes the check state of a button or check box control.
NTK_IsDlgButtonChecked(hwnd, nIDBtn) -> nVal (BST_*)	08.10.06	YES // Determines is a button control has a check mark next to it or whether a three-state button control is grayed, checked, or neither.
NTK_WinHelp(hwnd, cHelpFile, nCmd, cnData) -> lRet	12.10.06	YES // Call windows 32 bit help engine
DLL CALLS		
NTK_CallDll()	06.06.03	YES
NTK_GetProcAddress()	05.20.03	YES 06.06.03
NTK_FreeLibrary()	05.19.03	YES
NTK_LoadLibrary()	05.19.03	YES
NTK_LoadLibraryEx()	05.19.03	YES
DRAWING		
NTK_Arc()	09.01.03	YES
NTK_DrawEdge(hDc, aRect, nEdge, nFlags) --> lRet	25.03.05	YES
NTK_DrawFocusRect()	08.12.03	YES
NTK_DrawFrameControl(hDc, aRect, nType, nState) --> lRet	02.03.05	YES
NTK_Ellipse()	09.01.03	YES
NTK_ExtFloodFill()	09.01.03	YES
NTK_FillRect(hDC, nLeft, nTop, nRight, nBottom, hBrush) -> lRet	09.01.03	YES

NTK_FloodFill()	09.01.03	YES
NTK_FrameRect()	09.01.03	YES
NTK_GetPixel(hdc,nx,ny) -> nCurRgbClr	28.09.06	YES
NTK_LineTo()	08.11.03	YES
NTK_MoveTo()	08.11.03	YES
NTK_MoveToEx()	08.11.03	YES
NTK_Polyline()	02.11.04	YES
NTK_PolylineTo()	02.11.04	YES
NTK_Rectangle()	09.01.03	YES
NTK_RoundRect(hdc, nLeft,nTop,nRight,nBottom, nWellipse, nHellipse) -> lRet	09.01.03	YES
NTK_SetPixel(hdc,nx,ny,nRgbClr) -> nCurRgbClr	28.09.06	YES

FONTS

NTK_AddFontResource()	< 12.31.02	YES
NTK_ChoseFont(aFont, [@nPointSize], [nFlags], [@nColour],; [cStyle], [nFontType], [nMin], [nMax]) -> aFont or Nil	08.12.08	YES // Get the user to select a font
NTK_CreateFont()	< 12.31.02	YES
NTK_GetTextMetrics()	11.03.03	YES
NTK_RemoveFontResource()	< 12.31.02	YES
NTK_StringHeight(cString, hwnd, hFont) -> nHeight	08.08.05	YES
NTK_StringMaxSize(cString, hwnd, hFont) -> aSize = {nwidth,nHeight}	01.04.04	YES
NTK_StringSize(cString, hwnd, hFont) -> aSize = {nwidth,nHeight}	01.04.04	YES
NTK_StringWidth(cString, hwnd, hFont) -> nwidth	08.08.05	YES

FILES

NTK_GetFullPathName(cPathName,nLencPathBuf,@cPathBuf,@cFilePartBuf) -> nLen	01.25.06	YES	// retrieves the full path and file name of a specified file.
NTK_GetShortPathName(cLongName, @cShortNameBuf, nLencShortNameBuf) -> nLen	01.25.06	YES	// Obtains the short path form of a specified input path.
NTK_GetLongPathName(cShortName, @cLongNameBuf, nLencLongNameBuf) -> nLen	01.26.06	YES	// Obtains the Long path form of a specified input path.
NTK_FindFirstFile(cFileName, aw32FD) -> hFindFile	01.25.06	TESTING	// Searches a directory for a file whose name matches the specified filename. Use it when you want to know the longFileName of a ShortFileName
NTK_FindNextFile(hFindFile, aw32FD) -> lRet	01.25.06	TESTING	// Continues a file search from a previous call to the FindFirstFile function
NTK_FindClose(hFindFile) -> lRet	01.25.06	YES	// Closes the specified search handle previously obtained with FindFirstFile and eventually used with FindNextFile
NTK_IsDir(cDirPath) -> lRet	04.12.07	YES	// Is it a valid directory/folder?

GET SYSTEM

NTK_GetNew(aGetList, nGId, nGY, ngx,; nGHeight, nGWidth,; cGVar, cGCaption, cGPict,; cGType, hGFont,; hGFClr, hGBClr,; nGStyle, cGMsg,; bGPreVal, bGPostVal, bRange,; bGPick, cvarName, bonEvent, hwndP) -> aGetList			// Declare & Create GET
NTK_GetStart(aGetList, hwnd) -> lSuccess			// Start reading/running GETS
NTK_GetColor(aGetList,; hdcGet nwParam, hwndGet nlParam,; hDefBkgBrush) -> hRetBrush			// Colorize GETS
NTK_GetCtrl(aGetList, nwParam, nlParam) -> lSuccess			// MaskEdit realtime GET Control

GET SYSTEM'S API

NTK_ClearGets(aGetList) -> aGetList/{}			// Destroy all edits and Clear aGetList
NTK_ED_GetBuffer(hwndEdit) -> cBuffer			// Retrieve GET's current buffer
NTK_ED_GetCharPos(hwndEdit) -> nCharPos			// Retrieve the cursor position in the GET (in char number)
NTK_ED_GetCharPosEx(hwndEdit, [nCx], [nCy]) -> nCharPos			// Retrieve the cursor position in the GET (in char number)
NTK_ED_GetLen(hwndEdit) -> nMaxChar			// Retrieve GET's limit in chars
NTK_ED_GetLine(hwndEdit, nLine) -> cText			// Retrieve the text of a line in a memo/multiline GET
NTK_ED_GetLinePos(hwndEdit, [nCx], [nCy]) -> nLinePos			// Retrieve the current line position of the cursor in the GET
NTK_ED_GetSelInfo(hwndEdit) -> {nStartSel, nEndSel}			// Get's Selection Infos
NTK_ED_GetText(hwndEdit) -> cText			// Retrieve GET's text content
NTK_ED_KillChar(hwndEdit) -> Nil			// Clear last char sent to the GET
NTK_ED_ReplaceSel(hwndEdit, cNewTxt) -> Nil			// Replace current selection of the GET with the specified text
NTK_ED_SetCharPos(hwndEdit, nCharPos) -> Nil			// Move the cursor to the specified position with the GET (in char number)
NTK_ED_SetLen(hwndEdit, nMaxChar) -> Nil			// Define GET's max. char
NTK_ED_SetText(hwndEdit, cNewTxt) -> Nil			// Assign a new text to the GET
NTK_GetOriginalBuffer([nGetID]) -> xValue			// Return the previous value (C,N,D,L) stored into the current or specified GET
NTK_GiveGetActive([aGetList]) -> nCurGetID			// which GET has focus ?
NTK_GiveGetHandle([aGetList], [nGetID]) -> hwndEdit			// Return Edit handle of the asked GET
NTK_GiveGetType([aGetList], [nGetID]) -> cGetType			// ("U", GT_xxx or GXT_xxx constant)
NTK_IsGetEnable([aGetList], [nGetID]) -> lState			// Does the GET status is enable ?
NTK_IsGetUpdated([aGetList], [nGetID]) -> lState			// Does the GET has been modified ? nGetID=Nil or 0 means current Get.
NTK_LastGetEvent(void) -> nGetEvCode			// Returns the event code occurred to the current Get. This code is depending of Get type. (See EN_*, CB_*, LB_* or BN_* flags in

```

window.ch)
NTK_ReadExit( lShift ) -> lOldReadExit // Activate or not exit by Up,Dn or Enter keys
NTK_ReadInsert( lShift ) -> lOldReadIns // Switch between insert & overwrite mode
NTK_ReadModal( hWndParent, aGetList,; //
    nGetFocus, aBtnList,; // Get Reader. See READ Command.
    nRowMsg, nLeftColMsg, nRightColMsg,; // Same as Clipper + new GUI possibilities
    nMsgColor, hMsgFont,;
    nFocusRgbColor ) -> lGetUpdated
NTK_ReadEscape([hWndP]) -> Nil // Force to escape from current NTK_ReadModal() session
NTK_ReadValidate([hWndP]) -> Nil // Force to validate datas and exit from current NTK_ReadModal() session
NTK_ReadStandBy([lMode]) -> lOldMode // Force the current NTK_ReadModal() session to Pause (.T.) or to rePlay (.F.). If lMode is omitted, it returns the current
running mode.
NTK_RefreshGet(nGetID, [hWnd]) -> nil // Force Get(s) of a specified window to be refresh depending on nGetID: -1=Refresh all gets, 0=Nothing to do, nID=Get's ID to
refresh
NTK_SelectGetFocus( aGetList, nGetID ) -> nGetID // Change get position+focus or return current get focus...
NTK_SetBkgGetBrush( hBkgBrush ) -> hOldBrush // Set Bkg Gets Color Brush
NTK_SetGetGVK( nkey ) -> nOldGVK // Set the Global Validation Key for Get sessions
NTK_SetGetPickBmp(hPickBmp) -> hOldBmp // Set the default bitmap to use when the Get's PICK clause is declared. Default is OBM_COMBO.
NTK_SetGetState(aGetList, nGetID, lGetState ) -> lState // Enable/Disable a get in current aGetList
NTK_SetGetValue(aGetList, nGetID, xNewValue ) -> lUpdated // Modify/Change get value/var
NTK_SetSelGetBrush( hSelBrush ) -> hOldBrush // Set Selected Gets Color Brush. when get is active and owns the focus
NTK_SetUnselGetBrush( hUnselBrush ) -> hOldBrush // Set Unselected Gets Color Brush. when get is active, but has lost the focus
NTK_Updated() -> lUpdated // Does get(s) has/have been updated ?

```

GDI

```

NTK_BeginPaint() < 12.31.02 YES
NTK_CreateDC() 06.10.03 YES
NTK_CreateHatchBrush() 10.07.03 YES
NTK_CreatePatternBrush() 10.07.03 YES
NTK_CreatePen() 09.01.03 YES
NTK_CreateSolidBrush() 10.07.03 YES
NTK_DeleteDC() 06.10.03 YES
NTK_DeleteObject() < 12.31.02 YES
NTK_DeviceCaps() 09.01.03 YES
NTK_DrawIcon() 07.18.03 YES
NTK_DrawText() < 12.31.02 YES
NTK_EndPaint() < 12.31.02 YES
NTK_ExtTextout() < 02.11.04 YES
NTK_GetBkColor(hDC) -> nColor 28.07.05 YES
NTK_GetBkMode(hDC) -> nMode 28.07.05 YES
NTK_GetBValue(nRGB) -> nB 01.15.08 YES // Retrieves an intensity value for the BLUE component of a 32-bit red, green, blue
(RGB) value.
NTK_GetDC(hWnd) -> hDC 06.10.03 YES // Retrieves a handle of a display device context (DC) for the client area of the
specified window.
NTK_GetGValue(nRGB) -> nG 01.15.08 YES // Retrieves an intensity value for the GREEN component of a 32-bit red, green, blue
(RGB) value.
NTK_GetObject( hObject ) -> cInfo 07.20.03 YES // Get information about a specified graphics (GDI) object.
NTK_GetRValue(nRGB) -> nR 01.15.08 YES // Retrieves an intensity value for the RED component of a 32-bit red, green, blue
(RGB) value.
NTK_GetStockObject() < 12.31.02 YES
NTK_GetTextColor(hDC) -> nColor 28.07.05 YES
NTK_GetTextExtent(hDC, cString) -> nVal -> nW=Loword(nVal) nH=Hiword(nVal) 02.26.04 YES // Computes the width and height of the specified string of text.
NTK_GetTextExtentPoint32(hDC, cString) -> aVal -> {nWStr, nHStr} 02.26.04 YES // Computes the width and height of the specified string of text. This function
supersedes the NTK_GetTextExtent function.
NTK_GetWindowDC(hWnd) -> hDC 17.01.07 YES // Retrieves the device context (DC) for the entire window, including title bar,
menus, and scroll bars. A window device context permits painting anywhere in a window,
NTK_RGB(nR, nG, nB) -> nRGB < 12.31.02 YES // Blends the three red, green, blue (RGB) intensity components in order to make a
color that can be be output on GDI device.
NTK_ReleaseDC() 06.10.03 YES
NTK_SelectObject() < 12.31.02 YES
NTK_SetBkColor() < 12.31.02 YES
NTK_SetBkMode() < 12.31.02 YES
NTK_SetBrushOrgEx() 10.07.03 YES
NTK_SetTextAlign() < 02.11.04 YES
NTK_SetTextColor() < 12.31.02 YES
NTK_Textout() < 12.31.02 YES
NTK_UnRealizeObject() < 12.31.02 YES

```

INI FILES AND REGEDIT

```

NTK_GetProfileInt( <cSection>, <cEntryKeyName>, <nDefault> ) -> nValue 01.27.06 TESTING // Retrieves an integer from the specified key name in the given section of the
WIN.INI file
NTK_GetPvProfileInt(<cSection>, <cEntryKey>, <nDefault>, <cFile>) -> nValue 01.27.06 TESTING // Retrieves an integer associated with a key in the specified section of the given
initialization file
NTK_GetProfileString( <cSection>, <cEntryKeyName>, <cDefault> ) -> cValue 01.27.06 TESTING // Retrieves the string associated with the specified key in the given section of the

```

```

WIN.INI file
NTK_GetPVProfileString(<<Section>,<EntryKey>,<cDefault>,<cFile>) -> cValue 01.27.06 TESTING // Retrieves a string from the specified section in an initialization file.
NTK_WriteProfileString(<<Section>,<EntryKey>,<cString> ) -> lRet 01.27.06 TESTING // write a string into the specified section of the WIN.INI file.
NTK_WritePVProfileString(<<Section>,<EntryKey>,<cString>,<cFile>) -> lRet 01.27.06 TESTING // Write a string into the specified section of the specified initialization file

NTK_RegCloseKey( hKeyClose ) -> nRet 09.01.05 TESTING // Releases the handle of the specified key.
NTK_RegCreateKey( hKey, cSubKey ) -> nRet 09.01.05 TESTING // Creates the specified key. If the key already exists in the registry, the function
opens it
NTK_RegDeleteKey( hKey, cSubKey ) -> nRet 09.01.05 TESTING // Deletes a key and all its descendents.
NTK_RegEnumKeyEx( hKey, nIDSubkey ) -> nRet 09.01.05 TESTING // Enumerates subkeys of the specified open registry key. The function retrieves
information about one subkey each time it is called
NTK_RegOpenKeyEx( hKey, cSubKey ) -> nRet 09.01.05 TESTING // Opens the specified key.
NTK_RegQueryValueEx( hKey, cValueName ) -> nRet 09.01.05 TESTING // Retrieves the type and data for a specified value name associated with an open
registry key.
NTK_RegSetValueEx( hKey, cValueName, 0, nValueType, cData ) -> nRet 09.01.05 TESTING // Stores data in the value field of an open registry key.

IMAGES
NTK_AlphaBlend(hdcDst,nxDst,nYDst,nW,nH,;
hdcSrc, nXSrc, nYSrc, nWSrc, nHSrc, aBlendFunc ) -> lRet 04.30.07 TESTING // Displays bitmaps that have transparent or semitransparent pixels.
NTK_BitBlt(hdcDest,nXDest,nYDest,nWidth,nHeight,;
hdcSrc,nXSrc,nYSrc,nRop) -> lRet 07.19.03 YES // Perform a bit-block transfer of the color data corresponding to a rectangle of
pixels from the specified source device context into a destination device context.
NTK_cBmp2aBmp( cMemDib ) --> aBmpStruct 07.19.03 YES // Get the bitmap structure of a bitmap
NTK_CreateBitmap( nBmW,nBmH,nBmPlanes,nBmBitsPixel,cBmBits ) -> hBmp 07.19.03 YES // Create a bitmap that has the specified width, height, and bit pattern.
NTK_CreateCompatibleBitmap( hdc, nBmWidth, nBmHeight) -> hBmp 07.19.03 YES // Create a bitmap that is compatible with the given device.
NTK_CreateCompatibleDC( hdc ) -> hMemDC 07.21.03 YES // Create a memory device context that is compatible with the given device.
NTK_DrawBmp( hdc, hBmp, nX, nY, [nRop] ) -> nil 07.19.03 YES // Draw a bitmap using a specified raster op code. Default is SRCCOPY
NTK_DrawBmpTransparent( hdc, hBmp, nX, nY, [crTransp] ) -> nil 04.30.07 YES // Draw a transparent bitmap using a specified Rgb color as the transparency color.
Default is set to Magenta: NTK_RGB(255,0,255)
NTK_DrawTransparentBitmap( hdc, hBmp, nX, nY, [crTransp] ) -> nil 04.30.07 YES // Same as the above, just another syntax.
NTK_GetBmpRect( hBitmap ) -> aBmpRect 07.19.03 YES // Get the rectangle dimensions of a bitmap
NTK_GetDibRect( cDib ) -> aDibRect 07.19.03 YES // Get the rectangle dimensions of a bitmap previously stored/loaded into memory
with NTK_ReadDib().
NTK_GetObject( hObject ) -> cInfo 07.20.03 YES // Get information about a specified graphics (GDI) object.
NTK_LoadBitmap([<hInst>], <cnBmpRes> ) -> hBitmap 07.20.03 YES // Load a bitmap from a resource module (Executable or DLL fil
DLL file) or from the disk.
NTK_ReadBmp(cBmpFile) -> hBmp 07.19.03 YES // Load a bitmap file (*.bmp) from disk
NTK_ReadPictureToBmp(cImageFile) -> hBmp 17.03.06 YES // Load a image file (*.jpg, *.gif, *.bmp, *.wmf, *.ico) from disk
NTK_ReadDib(cDibFile) -> cDibBmp 07.18.03 YES // Load a Device Independent bitmap (*.bmp) from the disk
NTK_ReplaceBmpColor(hBmp, nOrgRgbColor, cNewRgbColor, [hDC]) -> hNewBmp 02.05.07 YES // Replace all pixel of a specified color by another color, in the bitmap.
NTK_ShowDib(<hDC>, <cDIB>, [nX], [nY], [ndwRop] ) -> NIL 07.21.03 YES // Draw in the specified context, a device independent bitmap previously
stored/loaded into memory with NTK_ReadDib().
NTK_StretchBlt(hdcDest,nXOriginDest,nYOriginDest,nWidthDest,nHeightDest,;
hdcSrc,nXOriginSrc,nYOriginSrc,nWidthSrc,nHeightSrc,;
nRop) -> lRet 07.19.03 YES // Copy a bitmap from a source rectangle into a destination rectangle, stretching or
compressing the bitmap to fit the dimensions of the destination rectangle, if necessary.
NTK_SetStretchBltMode(hdc, nStretchMode) -> nPrevStretchMode 07.19.03 YES
NTK_StretchDIBits(hdc, xDest, yDest, nWidthDest, nHeightDest,;
nXSrc, nYSrc, [nWidthSrc], [nHeightSrc],;
[acDIBits], [acDib], [nColor], [nRop]) -> nScanLines 09.14.08 TESTING // Copy a bitmap from a source rectangle into a destination rectangle,
stretching or compressing the bitmap to fit the dimensions of the destination rectangle, if necessary.
NTK_StoreBmpToMem( hBmp ) -> aBmpStruct 07.19.03 YES // Extract the structure of the specified bitmap and store it into a memory var.
NTK_TransparentBlt(hdcDest,nXOriginDest,nYOriginDest,nWidthDest,hHeightDest,; 04.30.07 YES // Perform a bit-block transfer of the color data corresponding to a rectangle of
pixels from the specified source device context into a destination device context.
hdcSrc,nXOriginSrc,nYOriginSrc,nWidthSrc,nHeightSrc,;
crTransparent) -> lRet

IMAGELIST
NTK_ImageList_Add(hIm1, hBmp, hBmpMask) -> nImgIdx 27.09.06 TESTING // Add an image or images to an image list.
NTK_ImageList_AddMasked(hIm1, hBmp, nRgbMask) -> nImgIdx 27.09.06 TESTING // Add an image or images to an image list, generating a mask from the
specified bitmap.
NTK_ImageList_Create( nCWidth, nCyHeight, nFlags, nInitial, nGrow)->hIm1 06.11.04 TESTING // Create a new image list.
NTK_ImageList_Destroy(hIm1) -> lRet 06.11.04 TESTING // Destroys an image list.
NTK_ImageList_DrawEx(hIm1, nIdx, hdcDest, nX, nY, nDX, DY, RgbBk, RgbFg, nStyle)->lRet 06.11.04 TESTING // Draw an image list item in the specified device context. The function
uses the specified drawing style and blends the image with the specified color.

```

NTK_ImageList_ExtractIcon(hi,hIml,nImgIdx) -> hIcon	27.09.06	TESTING	// Create an icon or cursor based on an image and mask in an image list.
NTK_ImageList_GetBkColor(hIml) -> nCurRgbBkg	27.09.06	TESTING	// Retrieve the current background color for an image list.
NTK_ImageList_GetImageCount(hIml) -> nImgCount	27.09.06	TESTING	// Retrieve the number of images in an image list.
NTK_ImageList_GetIcon(hIml,nImgIdx,nFlag) -> hIcon	27.09.06	TESTING	// Create an icon or cursor based on an image and mask in an image list.
NTK_ImageList_GetIconSize(hIml,@nCx,@nCy) -> lRet	06.11.04	TESTING	// Retrieve the dimensions of images in an image list. All images in an image list have the same dimensions.
NTK_ImageList_LoadBitmap(hInst,cBmpResName,nCx,nGrow,nRgbMask) -> hIml	27.09.06	TESTING	// Create an image list from the specified bitmap resource.
NTK_ImageList_LoadImage(hInst,cBmpResName,nCx,nGrow,nRgbMask,ntype,nFlag) -> hIml	27.09.06	TESTING	// Create an image list from the specified bitmap, cursor, or icon resource.
NTK_ImageList_ReplaceIcon(hIml,nIdx,hIcon) -> nImgIdx	27.09.06	TESTING	// Replace an image with an icon or cursor.
NTK_ImageList_SetBkColor(hIml,nRgbBkg) -> nPrevRgbBkg	27.09.06	TESTING	// set the background color for an image list
NTK_ImageList_SetIconSize(hIml,nCx,nCy) -> lRet	06.11.04	TESTING	// Set the dimensions of images in an image list and removes all images from the list.

KEYBOARD

NTK_GetAsyncKeyState(vKey) -> nVKStatus	12.28.03	YES
NTK_GetKeyState(vKey) -> nVKStatus	12.28.03	YES

HEADER & LISTVIEW

NTK_HD_Create(hwnd, nStyle, hMenu, hInst) -> hHDctrl		
NTK_HD_InsertItem(hHDctrl, nInsAfterItem, { cItemStr, hitmBmp, nitmFmt, nitmwidth, nitmHeight }) -> nRet		
NTK_HD_GetItem(hHDctrl, nColIdx, aHDItem) -> lRet // Pick-up infos from column header item, then store them into aHDItem[]...		
NTK_HD_SetItem(hHDctrl, nColIdx, aHDItem) -> lRet // Set a column header from infos previously stored into aHDItem[]. See aHDITEM[] members in NTKCTRL.CH		
NTK_HD_GetItemCount(hHDctrl)		
NTK_HD_GetItemRect(hHDctrl, nColIdx) -> aRect		

NTK_LV_CreateNubs(hwndHDR, nwidth, nheight) -> nRet		
NTK_LV_DeleteAllItems(hwnd) -> lRet		
NTK_LV_DeleteColumn(hwnd, nColIdx) -> Nil		
NTK_LV_EnsureVisible(hwnd, nColItemIdx, lPartialOk) -> lRet		
NTK_LV_GetBkColor(hwnd) -> nRGB		
NTK_LV_GetColumnWidth(hwnd, nColIdx) -> nwidth		
NTK_LV_GetItemCount(hwnd) -> nItemCount		
NTK_LV_GetItemRect(hwnd, nRowItemIdx, lvirCode) -> aRowItemRect		
NTK_LV_GetOutlineColor(hwnd) -> nBorderRgb		
NTK_LV_GetSelectedColumn(hwnd) -> nCol		
NTK_LV_GetSelectedItem(hwnd) -> nItem		
NTK_LV_GetSubItemRect(hwnd, nRowItemIdx, nColItemIdx, lvirCode) -> aCellItemRect		
NTK_LV_GetTextColor(hwnd) -> nRGB		
NTK_LV_GetViewRect(hwnd) -> aLVViewRect		
NTK_LV_InsertColumn(hwnd, nCol, cTitle, nwidth, nAlign, nImageListIdx) -> nNewColIdx or -1		
NTK_LV_InsertItem(hwnd, aRowItemColValue) -> nNewItemIdx or -1		
NTK_LV_ItemSetColor(nlParam, nRgbText, nRgbBkg) -> lRet		
NTK_LV_RedrawItems(hwnd, nFirstItem, nLastItem) -> lRet		
NTK_LV_Scroll(hwnd, ndx, ndy) -> lRet		
NTK_LV_SetBkColor(hwnd, nRGB) -> lRet		
NTK_LV_SetHeaderHeight(hwndHDR, nwidth, nheight) -> nRet		
NTK_LV_SetItemState(hwnd, nitem, nState, nMask) -> Nil		
NTK_LV_SetItemText(hwnd, nRowItemIdx, nColItemIdx, clabel) -> nil		
NTK_LV_SetOutlineColor(hwnd, nRgb) -> nOutlineRgb		
NTK_LV_SetTextBkColor(hwnd, nRGB) -> lRet		
NTK_LV_SetTextColor(hwnd, nRGB) -> lRet		
NTK_LV_SubItemHitTest(hwnd, nPtX, nPtY) -> aInfo -> {nRowIdx, nColIdx }		
NTK_LV_Update(hwnd, nItemIdx) -> lRet		

MENUS HI-LEVEL (RAD)

NtkAppendMenu(aMenu, cId, nFlags, cMenuItem, bAction, chlpmsg) -> lRet	< 04.30.03	YES
NtkCreateMenu(void) -> aMenu	< 04.30.03	YES
NtkCreatePopupMenu(void) -> aPopupMenu	< 04.30.03	YES
NtkEnableMenuItem(aMenu, cnId, nFlag) -> nState //MF_... or -1 (Err)	31.12.05	YES
NtkGetMenu(hwnd) -> aMenu	< 04.30.03	YES
NtkGetMenuBlock(aMenu, cnMenuItem) -> bMenuItem	< 04.30.03	YES
NtkGetMenuHandle(aMenu, [cnId]) -> hMenu hPopupMenu	01.03.06	YES
NtkGetMenuHlpMsg(aMenu, cnMenuItem) -> cMenuHlpMsg	< 04.30.03	YES
NtkSetMenu(hwnd, aMenu) -> lRet	< 04.30.03	YES
NTKTrackPopup(aMenu, nFlags, x, y, hwnd, [aDismissRect Nil]) -> lRet	01.03.06	YES

MENUS LO-LEVEL

NTK_AppendMenu(hMenu, nFlags, nCID, cMenuItem) -> lRet	< 04.30.03	YES
NTK_CheckMenuItem(hMenu, nItem, nChkFlag) -> nPrevState	28.09.06	TESTING // Set the state of the specified menu item's check mark attribute to either checked

or unchecked.

NTK_CreateMenu(void) -> hMenu	< 04.30.03	YES	
NTK_CreatePopupMenu(void) -> hMenu	< 04.30.03	YES	
NTK_DestroyMenu(hMenu) -> lRet	01.03.06	YES	// Destroy the specified menu and frees any memory that the menu occupies.
NTK_GetMenu(hwnd) -> hMenu	01.03.06	YES	// Retrieve the handle of the menu assigned to the given window.
NTK_GetMenuCheckMarkDimensions(void) -> nwidth_nHeight	28.09.06	YES	// Return a value which is both the width and the height of the menu check mark. Use
NTK_Loword() to extract width, and NTK_Hiword() for the height.			
NTK_IsMenu(hMenu) -> lRet	< 04.30.03	YES	
NTK_LoadMenu(hInst, cMenuResNamle) -> hMenu	08.09.08	YES	// Load the specified menu resource from the executable (.EXE) file associated with an
application instance.			
NTK_SetMenu(hwnd, hMenu) -> lRet	< 04.30.03	YES	
NTK_SetMenuItemBitmaps(hMenu, nPos, nFlag, hBmpUnChk, hBmpChk) -> lRet	28.09.06	TESTING	// Associate the specified bitmap with a menu item. whether the menu item is checked
or unchecked, windows displays the appropriate bitmap next to the menu item.			
NTK_TrackPopupMenu(hMenu, nFlags, x, y, hwnd, [aDismissRect Nil]) -> lRet	01.03.06	YES	

MESSAGES

NTK_DispatchMessage(aMsg) -> nRet	< 12.31.02	YES	
NTK_IsDialogMessage(hwnd, aMsg) -> lRet	06.23.03	YES	
NTK_GetMessage(aMsg, hwnd, nMin, nMax) -> lRet	< 12.31.02	YES	
NTK_PeekMessage(aMsg, hwnd, nMin, nMax, nRemoveFlag) -> lRet	< 12.31.02	YES	
NTK_PostMessage(hwnd, nMsg, nwParam, nlParam) -> nRet	< 12.31.02	YES	07.01.03
NTK_PostQuitMessage(nwParam) -> Nil	< 12.31.02	YES	
NTK_SendMessage(hwnd, nMsg, nwParam, nlParam) -> nRet	< 12.31.02	YES	07.01.03
NTK_TranslateAccelerator(hwnd, hAccel, aMsg) -> lRet	< 12.31.02	YES	
NTK_TranslateMDISysAccel(hwnd, aMsg) -> lRet	08.14.08	YES	vPro Only
NTK_TranslateMessage(hwnd, aMsg) -> lRet	< 12.31.02	YES	

MISCELLANEOUS

NTK_DestroyIcon()	07.18.03	YES	
NTK_GetInstance()	< 12.31.02	YES	
NTK_GetLastError()	05.19.03	YES	
NTK_GetModuleFileName([hInst]) --> cPathFileName	08.01.05	YES	// Retrieves the full path and filename for the executable file containing the
specified module.			
NTK_GetModuleHandle(cModuleName) --> nRet	01.23.06	YES	// Returns a module handle for the specified module if the file has been mapped into
the address space of the calling process.			
NTK_GetSysColor(nIndex) -> nRGB	05.19.03	YES	// Retrieves the current color of the specified display element
NTK_GetSysColorBrush(nIndex) -> hBrush	05.19.03	YES	// Retrieves a handle identifying a logical brush that corresponds to the specified
color index.			
NTK_GetSystemMetrics()	04.01.04	YES	
NTK_GetVersion(void) -> NOSVERSION	08.26.06	YES	// Returns the current version number of windows and information about the OS that is
currently running.			
NTK_GetVersionEx(void) -> aOSVERSIONINFO	08.26.06	YES	// Obtains extended information about the version of the OS that is currently
running.			
NTK_InitCommonControls()	08.12.03	YES	
NTK_InitCommonControlsEx()	08.12.03	YES	
NTK_LoadIcon()	< 12.31.02	YES	
NTK_LoadResource()	< 12.31.02	YES	
NTK_MakeIntResource()	< 12.31.02	YES	
NTK_MessageBeep(nSound)	06.07.04	YES	
NTK_Rand([nRandLim]) -> nRandNumber (if nRandNumber<0 => Error)	11.01.08	YES	// Generates random numbers between range [0,nRandLim]. Note: nRandLim <=MAX_RANDOM
(32767=0x7fff)			
NTK_SetHandleCount()	07.19.03	YES	
NTK_SetLastError()	05.19.03	YES	
NTK_WaitForSingleObjectEx(hHandle, nMilliSecs, bAlertable) -> nwaitCode	02.01.06	YES	// Forces the OS to wait until an application has been terminated.
NTK_Version(void) -> cNTKversion	08.26.06	YES	// Returns the current version of NTK project as a string

MOUSE

NTK_GetCapture(void) -> hwndCapture	03.29.05	YES	
NTK_GetReleaseCapture(void) -> lRet	03.29.05	YES	
NTK_SetCapture(hwnd) -> hPrevwnd	03.29.05	YES	
NTK_TrackMouseEvent(hwndToTrack, nHoverFlags, nHoverTime) -> lRet	12.21.05	YES	// posts msg when the mouse pointer leaves a window or hovers over a
window for a specified amount of time			

PRINT

NTK_AbortDoc(hDC) -> nStatus	08.31.06	YES	// Stop the current print job and erases everything drawn since the
last call to the StartDoc function.			
NTK_EndDoc(hDC) -> nStatus	08.31.06	YES	// Stops a print job.
NTK_EndPage(<hDC>) -> nStatus	08.31.06	YES	// Inform the device that the application has finished writing to a
page. Used to direct the device driver to advance to a new page.			
NTK_GetPrintDC(void) -> hPrintDC	08.31.06	YES	// Display a Print dialog box and returns a printer DC for drawing

purposes			
NTK_PrintDlg(aPrintDlg) -> nRet (.T., .F. or Nil)	08.31.06	YES	// (input/output) Displays a Print dialog box and allow to set up printer and print job properties.
NTK_StartDoc(hDC,cDocName,[cOutput]) -> nStatus	08.31.06	YES	// Start a print job.
NTK_StartPage(<hDC>) -> nStatus	08.31.06	YES	// Prepare the printer driver to accept data.
RAD SYSTEM APIS			
NTK_AutoCheckEvent(aBtnList, hWnd, hAccel, UsrEvtProc) -> Nil			
NTK_SendCloseEvent([hWnd]) -> nRet			// Force closing current window hWnd
NTK_SendQuitEvent(Void) -> always .T.			// Force to quit the whole application
__NtkBoxTo(hWnd, hDC, nTop, nLeft, nBottom, nRight, cColor, nStyle) -> nil			
__NtkDefSayFont([hFont]) -> hOldDefFont			
__NtkGetMDIClient(hWndFrame) -> hWndMDIClient	08.19.08	YES	// Retrieve the handle to the MDIClient window of the specified MDIFRAME parent container.
__NtkMakeWnd(lMDI, nStyle, hInst, hIcon, hCurs, hBrush, cClass, nWndExStyle, cWndTitle, nWndStyle, nX, nY, nW, nH, hWndP, hMenu, bOnInit, bOnExit, bOnPaint, bOnMsg) -> hWnd			
__NtkMaxCol([hWndP]) -> nWndMaxCol			
__NtkMaxRow([hWndP]) -> nWndMaxRow			
__NtkMenuMsgSettings(nRow, nStyle, hMenuMsgFont, nRgbColor, hWndP) -> aOldArea: {nLeft,nTop,nRight,nBottom}			
__NtkSay([hWnd], [hDC], nRow, nCol, [nHeight], [nWidth], xSayExpr, [cPic], [fgColor], [bgColor], [nStyle], [hFont]) -> nil			
__NtkSetColRatio([nRatio]) -> nOldRatio			
__NtkSetColor([cColors]) -> cCurColors			
__NtkSetPixMode([lMode]) -> lOldMode			
__NtkSetRGBColor(nRgbFg, nRgbBg) -> aOldColors: {nOldRgbFg, nOldRgbBg}			
__NtkSetRowRatio([nRatio]) -> nOldRatio			
__NtkTBRowDb(cnAlias, aColNames, aColTitles, aColAlign, aColSizes, hWndP, nId, nTop,nLeft,nBottom,nRight, nStyle, hFont, aSettings, bOnCfg, bOnValid, lRun) -> oTB			// An easy way to create and display a NtkTBrowse child control
REGIONS			
NTK_CreateRectRgn(nLeftRect,nTopRect,nRightRect,nBottomRect) -> hRgn	10.17.06	YES	// Creates a rectangular region.
NTK_CreateRectRoundRgn(nLeft,nTop,nRight,nBottom,nWidthEllipse,nHeightEllipse) -> hRgn	10.17.06	YES	// Creates a rectangular region with rounded corners.
NTK_GetWindowRgn(hWnd,hRgn) -> nTypeRgn	10.17.06	YES	// Obtains a copy of the window region of a window. The window region determines the area within the window where the O.S. permits drawing.
NTK_SetWindowRgn(hWnd,hRgn,lRedraw) -> lRet	10.17.06	YES	// Sets the window region of a window. The O.S. does not display any portion of a window that lies outside of the window region .
RICHEDIT APIS			
NTK_RE_GetCharFormat(hRedit, aCharFormat2, [nFmtRange]) -> lRet	06.29.08	YES	// Determines text formatting of the whole or selected portion of a richedit.
NTK_RE_LoadRtfFile(hRedit, cFileName) -> lRet	06.29.08	YES	// Loads datas from a RTF standard filedisk and store them into the current richedit.
NTK_RE_PrintRTF(hRedit, hDC/hdcPrinter) -> lRet	06.29.08	TESTING	// Print the specified RicheEdit content.
NTK_RE_SaveRtfFile(hRedit, cFileName) -> lRet	06.29.08	YES	// Saves RichEdit datas as a filedisk using the standard word processor RTF format.
NTK_RE_SetCharFormat(hRedit, aCharFormat2, [nFmtRange]) -> lRet	06.29.08	YES	// Formats the whole or selected portion of text within the current richedit.
SHELLEXEC, DISK AND DIRECTORY			
NTK_Createdirectory(cPathName) -> lRet	12.30.05	YES	// Creates a new directory with the default security descriptor of the calling process
NTK_GetCurrentDirectory(void) -> cPathName	12.30.05	YES	// Retreives the current directory for the current process.
NTK_GetSystemDirectory(void) -> cPathName	12.30.05	YES	// Retrieves the path of the windows system directory
NTK_GetWindowsDirectory(void) -> cPathName	12.30.05	YES	// Retrieves the path of the windows directory
NTK_Run(cExeFile, cParams, lWait, cDefDir, nCmdShow, hWndP) -> nRetCode (0-32)	02.01.06	YES	// Execute an other application from the current
NTK_Removedirectory(cPathName) -> lRet	12.30.05	YES	// Deletes an existing empty directory
NTK_SHBrowseForFolder(hWndP,cPidlRoot cTitle,nFlags,nIdxImage) -> cPIDL	02.01.06	YES	// See common dialoge section
NTK_SetCurrentDirectory(cPathName) -> lRet	12.30.05	YES	// Changes the current directory for the current process.
NTK_ShellExeute(hWndP,cOperation,cFileName,cExeParams,cDefDir,nShowCmd) -> hWndApp or nErr (0-32)	02.01.06	YES	// Opens or prints a specified file. The file can be an executable file or a document file
NTK_ShellExeuteEX(aSHExecInfo) -> lRet. After call, check for aSHExecInfo array members a document.	02.01.06	YES	// Performs an action on a file. The file can be an executable file or a document.
NTK_WinExec(cCmdLine, nShowCmd) -> nSuccess (>31) or nErr (0-31)	02.01.06	YES	// Runs the specified application
STRINGS			
NTK_OemToAnsi()	< 12.31.02	YES	
NTK_AnsiToOem()	< 12.31.02	YES	
NTK_MultiByteToWideChar(nCPage,nFlags,cStr2conv) -> cWideStr	17.01.07		// Maps a character string to a wide-character (Unicode) string.

NTK_WideCharToMultiByte(ncPage,nFlags,cWideStr2conv) -> cStrConv

17.01.07 // Maps a wide-character string to a new character string.

STRUCTURE TO ARRAY

NTK_A2Rect(aRect) -> rc (C RECT structure)
NTKGDI.CH)
NTK_DrawItem2A(nlParam) -> aDIS[]
NTKGDI.CH)
NTK_NMCustomDraw2A(nlParam) -> aNMC[]
also NTKCCTRL.CH)
NTK_NMHDR2A(nlParam) -> aNMHDR[]
NTKCCTRL.CH)
NTK_NMLListView2A(nlParam) -> aNMLV[]
NTKCCTRL.CH)
NTK_NMLVCustomDraw2A(nlParam) -> aNMLVCD[]
also NTKCCTRL.CH)
NTK_MinMaxInfo2A(nlParam)
(see also WINDOWS.CH)
NTK_Rect2A(rc) -> aRect
NTKGDI.CH)
NTK_Str2Ptr(String) -> LONG_PTR

08.20.07 YES // Convert an [x]Harbour array to a C RECT structure. (see also
< 12.31.04 YES // Convert a C NMHDR structure to array. See WM_DRAWITEM . (see also
< 12.31.04 YES // Convert a C NMCUSTOMDRAW structure to array. See WM_NOTIFY. (see
< 12.31.04 YES // Convert a C NMHDR structure to array. See WM_NOTIFY. (see also
< 12.31.04 YES // Convert a C NMLISTVIEW structure to array. See WM_NOTIFY. (see also
< 12.31.04 YES // Convert a C NMLVCUSTOMDRAW structure to array. See WM_NOTIFY. (see
08.20.07 TEST // Convert a C MINMAXINFO structure to array. See WM_GETMINMAXINFO
08.20.07 YES // Convert a C RECT structure to an [x]Harbour array. . (see also
< 12.31.04 YES // Convert a Clipper/xHrb string to a (LONG) C pointer

TERMINAL

NTK_ClearTypeAhead(Void) -> Nil
NTK_Inkey(nTempo, aMsg) -> nLastKeyStroke
Pump)
NTK_LastBtn() -> nBtnID
NTK_LastKey() -> nKey
NTK_MouseState() -> nMseBtnState
btns are UP
NTK_MouseCol() -> nColPos
NTK_MouseRow() -> nRowPos
NTK_MouseX() -> nXPos
NTK_MouseY() -> nYPos
NTK_Mouse(nX0, nX1, nY0, nY1) -> lRet
depending on SET PIXEL state)
NTK_OkMouse(nX0, nX1, nY0, nY1) -> lRet
(The area can pixel or row/col depending on SET PIXEL state)
NtkInkey([nwait], [hAcce], [hwnd]) -> nKeyStroke
Ntk_SendVKey(nVK, [hwnd]) -> nRet
the specified window

< 12.31.04 YES // Reset respectively keyboard, mouse and lastbtn buffer
< 12.31.04 YES // Must be intragated into a DO WHILE GetMessage() loop (Event+Message
< 12.31.04 YES // Returns the ID of the last button activated
< 12.31.04 YES // Returns the K_* value of the last keystroke
< 12.31.04 YES // 1=Left Btn is pushed 2=Right Btn is pushed -1=Double Click 0=All
< 12.31.04 YES // Screen Mouse Col position. Depends from SET COL RATIO ... settings
< 12.31.04 YES // Screen Mouse Row position. Depends from SET ROW RATIO ... settings
< 12.31.04 YES // Screen (pixel) Mouse X position.
< 12.31.04 YES // Screen (pixel) Mouse Y position.
< 12.31.04 YES // Is mouse over the specified area ? (The area can pixel or row/col
< 12.31.04 YES // Is mouse over the specified area, and Mouse Left Btn is pushed ?
< 12.31.04 YES // Creates a waitstate for keyboard & Mouse events
< 12.31.04 YES // Emulates a virtual keystroke by generating a WM_KEYDOWN event to

TIMER

NTK_KillTimer(hwnd, hTimer) -> lRet
NTK_SetTimer(hwnd, nIdTimer, nMillisecs) -> hTimer
NTK_SetTimerEx(hwnd, nIdTimer, nMillisecs, BUdFAction) -> hTimer
NTK_GetTickCount(void) -> nMillisecs
was started.

05.19.03 YES
05.19.03 YES
05.19.03 YES
11.05.07 YES // Rieves the number of milliseconds that have elapsed since windows

THUMBS & SCROLL BARS

NTK_GetScrollInfo(hwnd,nBar,aScrollInfo) -> lRet
NTK_GetScrollPos(hwnd, nBar) -> nRet
NTK_GetScrollRange(hwnd, nBar, @nMinPos, @nMaxPos) -> lRet
NTK_SetScrollInfo(hwnd,nBar,aScrollInfo,lRedraw) -> nOldSBCurPos
NTK_ScrollWindow(hwnd,nXAmout,nYAmout,aRectScroll, aRectClip) -> lRet
NTK_SetScrollPos(hwnd, nBar, nPos, lRedraw) -> nOldSBCurPos
NTK_SetScrollRange(hwnd, nBar, nMinPos, nMaxPos, lRedraw) -> lRet
NTK_ScrollDC(hdc,nXAmout,nYAmout,aRectScroll,
aRectClip,hRgnUpd,aRectUpd) -> lRet
NTK_ShowScrollBar(hwnd,nSBFlag,lShow) -> lRet

11.05.04 YES
11.05.04 YES
11.05.04 YES
11.05.04 YES
11.05.04 YES
11.05.04 YES
11.05.04 YES
11.06.04 YES
11.07.04 YES

TREE VIEW

NTK_TV_GetItem(hwndTV, aTVI) -> lRet
NTK_TV_GetParent(hwndTV, hItem) -> hTreeItem
NTK_TV_GetRoot(hwndTV) -> hTreeItem
NTK_TV_GetSelection(hwndTV) -> hTreeItem
NTK_TV_InsertItem(hwndTV, hParentTree, hInsAfter, nItemMask, hItem, nItemState, nItemMask, cItemText, nIdxUnSelImg, nIdxSelImg, nHasChild, nlParam) -> hTreeItem
NTK_TV_SelectDropTarget(hwndTV, hItem) -> lRet
NTK_TV_SetItem(hwndTV, nItemMask, hItem, nItemState, nItemMask, cItemText, nIdxUnSelImg, nIdxSelImg, nHasChild, nlParam) -> lRet

```

WINDOWS
NTK_BringWndToTop() < 12.31.02 YES
NTK_ClientToScreen() < 12.31.02 YES
NTK_CallWindowProc(<nProc>, <hwnd>, <nMsg>, <nwParam>, <nIParam>) -> nResult 10.20.03 YES
NTK_CenterWindow(hwnd, [hwndParent]) -> {nx,ny,nxSize,nySize} 04.04.06 YES // Center a child window in a parent window. Default is the whole
screen.
NTK_CloseWindow(hwnd) -> lRet 08.26.06 YES // Convert an window to an icon
NTK_CreateWindow() < 12.31.02 YES
NTK_CreateWindowEx() < 12.31.02 YES
NTK_CreateToolTip(nStyle, cTTMsg, nLeft, nTop, nRight, nBottom, hwndP, nuFlags, nuId, hInst) -> hwndTT 12.21.05 YES // Create a tooltip window
NTK_CreateMLToolTip(nStyle, cTTMsg, nLeft, nTop, nRight, nBottom, hwndP, nuFlags, nuId, hInst) -> hwndTT 02.28.06 YES // Create a multiline tooltip window
NTK_CreateUpDownControl( nStyle, nx, ny, nwidth, nHeight, hwndP,;
nID, hInst, hwndBuddy, nMaxRange, nMinRange, nPos) -> hwndUpDn 04.13.06 YES // Create a UpDown Control (Spinner)
NTK_DefWindowProc(hwnd, nMsg, nwParam, nIParam) -> nResult < 12.31.02 YES < 12.31.02 YES
NTK_DefWndProc(hwnd, nMsg, nwParam, nIParam) -> nResult < 12.31.02 YES
NTK_DefFrameProc(hwnd, hwndMDIClient, nMsg, nwParam, nIParam) -> nResult 08.14.08 YES // vPro Only
NTK_DefMDIChildProc(hwnd, nMsg, nwParam, nIParam) -> nResult 08.14.08 YES // vPro Only
NTK_DestroyWindow() < 12.31.02 YES
NTK_EnableWindow() < 12.31.02 YES
NTK_FindWindow() < 12.31.02 YES
NTK_GetActiveWindow() < 12.31.02 YES
NTK_GetClassName(hwnd) -> cClassName < 07.31.06 YES // Retrieves the Class Name of a window
NTK_GetClientRect(hwnd) -> aClientRect < 12.31.02 YES
NTK_GetDesktopWindow() < 12.31.02 YES
NTK_GetFocus(hwnd) -> hwndFocused 05.19.03 YES
NTK_GetForegroundWindow() 05.19.03 YES
NTK_GetParent(hwnd) -> hwndParent 10.13.03 YES
NTK_GetWindow() < 12.31.02 YES
NTK_GetWindowLong() < 12.31.02 YES
NTK_GetWindowRect(hwnd) -> aRect < 12.31.02 YES
NTK_GetWindowText(hwnd) -> cTitle 10.21.03 YES
NTK_InvalidateRect(hdc, aRect, lEraseBkg) -> lRet < 12.31.02 YES
NTK_InvertRect(hdc, aRect) -> lRet 11.05.04 YES
NTK_IsChild() < 12.31.02 YES
NTK_IsIconic() < 12.31.02 YES
NTK_IsWindow() < 12.31.02 YES
NTK_IsWindowEnabled() < 12.31.02 YES
NTK_IsWindowVisible() < 12.31.02 YES
NTK_IsZoomed() < 12.31.02 YES
NTK_HideWindow(hwnd, [lState]) -> Nil 09.05.07 YES // Hide or recall a window from/to the screen (depending on lState
flags)
NTK_MapWindowPoints(hwndSrc, hwndDest, nPtX, nPtY) -> nPtXY (Loword,Hiword) 04.14.05 YES
NTK_MoveWindow(hwnd, nx, ny, nw, nh, lRepaint) -> lRet < 12.31.02 YES
NTK_RedrawWindow(hwnd, aRect, hRegion, nFlags) -> lRet 22.05.04 YES
NTK_RegisterClass(nStyle, hInst, hIcon, hCursor, hBgBrush,;
cClass, bwndProcName, [cMenuName],;
[ncbClsExtra], [ncbwndExtra]) -> lRet < 12.31.02 YES
NTK_RegisterClassEx(nStyle, hInst, hIcon, hCursor, hBgBrush,;
cClass, bwndProcName, [cMenuName],;
[ncbClsExtra], [ncbwndExtra], [hIconSm]) < 12.31.02 YES
NTK_ScreenToClient() < 12.31.02 YES
NTK_SelectWindow(hwnd) -> hwndCur < 12.31.02 YES
NTK_SetFocus(hwnd) -> hwndPrev 05.19.03 YES
NTK_SetForegroundWindow(hwnd) -> lRet 05.19.03 YES
NTK_SetParent() 10.13.03 YES
NTK_SetWindowLong() < 12.31.02 YES
NTK_SetWindowOrgEx(hdc, x, y, aPt) -> lRet 05.13.05 YES
NTK_SetWindowPos() 06.26.03 YES
NTK_SetWindowText() 10.21.03 YES
NTK_ShowWindow() < 12.31.02 YES
NTK_SubClassWindow(<hwnd>, <bNewWndProcName>, [xReserved]) --> nProc 10.20.03 YES
NTK_UnRegisterClass() < 12.31.02 YES
NTK_UpdateWindow() < 12.31.02 YES
NTK_ValidateRect(hdc, aRect) -> lRet 11.06.04 YES

```