

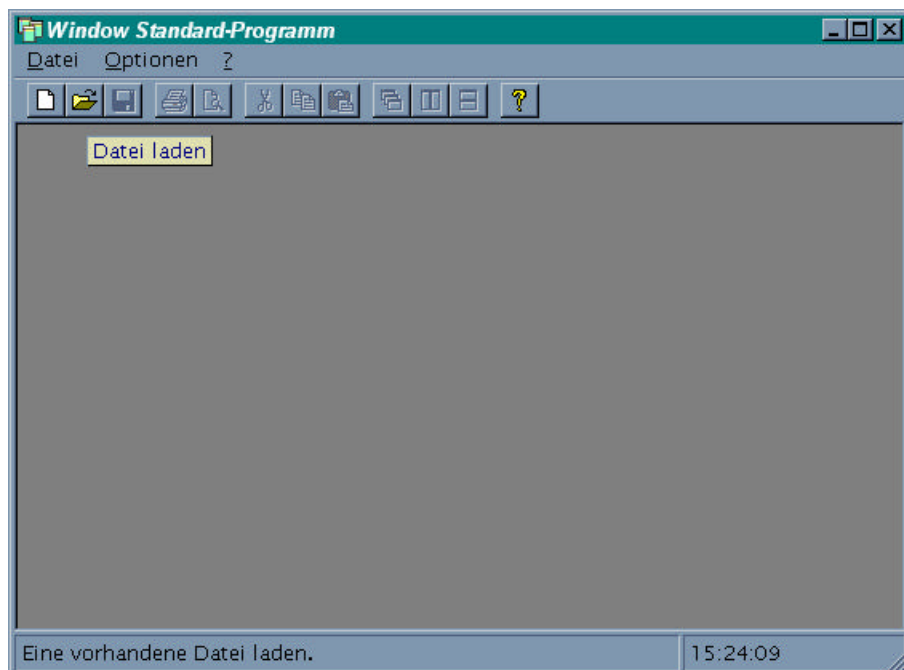
Die Bibliothek für GFA-Basic

zu

Windows 95[®], Windows NT[™] und Windows[®] 3.11

CTRLTOOL.DLL

Funktionen und ihre Meldungen



Eberhard Funck

1997

Danksagung:

Für die Schaffung des GFA Basic für Windows an:

GFA Systemtechnik GmbH

Für die Anregung, lange Dateinamen in GFA-Basic zu verwenden und damit den Grundstein für eine FileSelectBox für alle Systeme in Windows 95-Look zu verwirklichen an:

Roland Walter, D-10437 Berlin

Für die Anregung, die Funktionen der 32 Bit DLLs im GFA Basic verwenden zu können:

(c) Sjouke Hamstra, GFA Intermediary

Dipl.-Ing. Eberhard Funck
Unter den Eichen 14
31139 Hildesheim

Inhalt

InitCtrlTool()	Seite 5
CreateStatusbar()	Seite 6
SB_SETPARTS	
SB_SETTEXT	
CreateToolbar()	Seite 9
TB_ADDBITMAP	
TB_ADDBUTTONS	
TB_ADSTRING	
TB_AUTOSIZE	
TB_BUTTONSTRUCTSIZE	
TB_COMMANDTOINDEX	
TB_ENABLEBUTTON	
TB_GETITEMRECT	
TB_GETSTATE	
TB_GETTOOLTIPS	
TB_SETBITMAPTOMENU	
WM_NOTIFY	Seite 16
TTN_NEEDTEXT	
TTN_SHOW	
TTN_POP	
TTM_ADDTOOL	
MenuHelp()	Seite 20
SendSpinMessage()	Seite 21
SendKarteiMessage()	Seite 23
Progress Bar Messages	Seite 25
PBM_DELTAPOS	
PBM_SETCOLOR	
PBM_SETPOS	
PBM_SETRANGE	
PBM_SETSTEP	
PBM_STEPIT	

OpenLongFileName()	Seite 29
SaveLongFileName()	Seite 30
PrinterSetup()	Seite 32
GetPrinterOrientation()	Seite 33
SetPrinterOrientation()	Seite 33
SystemInfo()	Seite 34
Ctl3DOn()	Seite 35
Ctl3DOn()	Seite 35
Ctl3DCtlOn()	Seite 35
CopyBmp()	Seite 36
GetString()	Seite 37
ChangeString()	Seite 38
GetSystemTime()	Seite 39
DayOfWeek()	Seite 39
MoonAge()	Seite 40
CopyFile()	Seite 40
DeleteFile()	Seite 41

BOOL InitCtrlTool ()

Die InitCtrlTool()-Funktion muß deklariert und vor allen anderen Funktion aufgerufen werden. Der Aufruf dieser Funktion stellt sicher, daß alle Fensterklassen der CTRLTOOL.DLL initialisiert werden. Wird dies vergessen, werden alle Fensteraufrufe ignoriert.

Declaration:

```
DLL #1, "CTRLTOOL"  
DECL BOOL InitCtrlTool ()  
ENDDLL
```

Parameter	Beschreibung	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
Keine	Keine			
Aufruf	~^InitCtrlTool () Folgende eigenständige Fensterklassen werden durch Aufruf dieser Funktion global initialisiert: statusbar_class toolbar_class tooltip_class progressbar_class kartei_class spin_class			

WORD CreateStatusbar (style%, lpsz%, hwnd, id)

LONG style% /* Statusbar-Stil */
LONG lpsz% /* Zeiger auf einen String */
WORD hwnd& /* Fensterhandle */
WORD id& /* Identifikations-Nummer */

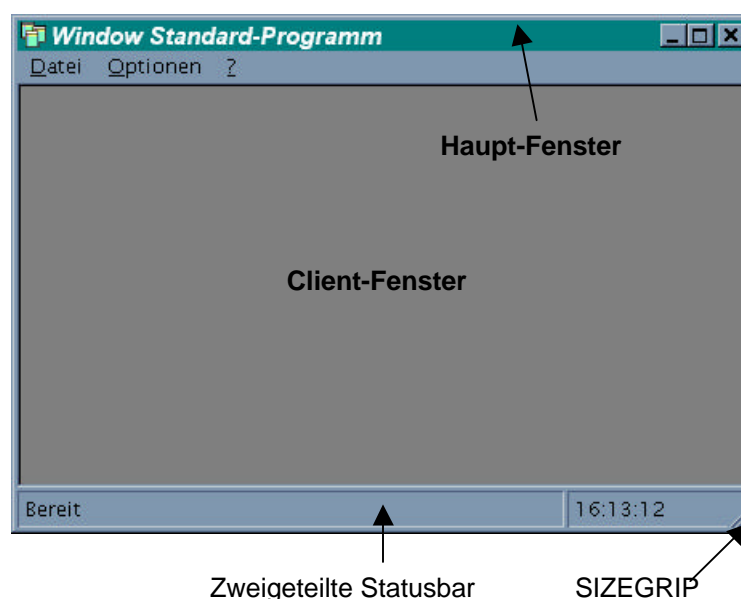
Die CreateStatusbar-Funktion integriert eine Statusbar-Leiste nach WIN95-Art am unteren Fensterrand. Diese Funktion ist identisch mit der *CreateStatusWindow()-Funktion*, die in der COMMCTRL.DLL (16 Bit-Version), sowie der COMCTL32.DLL (32 Bit-Version) eingebunden ist. Damit Programme mit Statusbar-Windows auch unter Windows NT laufen, hier fehlt die 16 Bit-DLL, ist diese Funktion mit den notwendigsten Meldungen nachempfunden worden. Mit der CreateWindow()-Funktion und der "statusbar_class"-Fensterklasse kann die Statusbar ebenfalls aufgerufen werden.

Declaration:

```
DLL #1, "CTRLTOOL"  
DECL WORD CreateStatusbar (l,l,w,w)  
ENDDLL
```

Parameter	Beschreibung	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
style%	Fensterstile der Statusbar mit "ODER" verknüpft. WS_CHILD WS_VISIBLE muß immer gesetzt werden. SBARS_SIZEGRIP = \$100, kann mit dem WS-Stil geodert (!) werden. Einbindung der Riffelung in der rechten unteren Fensterecke zur Vergrößerung oder Verkleinerung des Fensters (auch unter Windows 3.11)			
lpsz%	Zeiger auf einen null-terminierten String, der im ersten Abschnitt der Statusbar erscheint.			
hwnd&	Handle des Parent-Windows, zu der die Statusbar-Leiste gehört.			
id&	Identifikations-Nummer der Statusbar. Wird für die Meldungen benötigt, die an das Mutterfenster gesendet werden.			
Rückgabe:	Bei Erfolg das Handle der Statusbar, bei Mißerfolg wird NULL zurückgeliefert.			

Beispiel eines Standard Window-Fensters mit Statusbar und SIZEGRIP:



Statusbar-Message:

Die beiden folgenden Statusbar Message-Variablen müssen zuvor im Programm initialisiert werden.

SB_SETPARTS	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
Wert	WM_USER + 4		
Beschreibung	SB_SETPARTS ist eine Meldung, die die Anzahl der Abschnitte in der Statusbar und die Koordinaten der rechten Ecken jedes Abschnittes festlegt.		
wParam&	Anzahl der Abschnitte, die in der Statusbar dargestellt werden sollen.		
lParam%	Zeiger auf ein WORD-Array, das die gleiche Anzahl von Feldern wie in wParam& besitzen muß. Das letzte Array wird auf -1 gesetzt, da das rechte Ende des letzten Abschnitts immer gleich der rechten Fenster-Koordinate ist (Ende-Erkennung).		
Rückgabe:	BOOL: wenn erfolgreich, TRUE wird zurückgegeben, sonst FALSE		

Beispiel: Statusbar in zwei Abschnitte teilen (siehe Bild oben)

```
DIM statusWidths&(1)
~GetClientRect (hwnd&, *rect.)
statusWidths&(0) = 3 * rect.right / 4
statusWidths&(1) = -1
~SendMessage (hwndStatusbar, SB_SETPARTS, 2, V:statusWidths&(0))
```

Der erste Abschnitt in der Toolbar ist 3/4 der Gesamtlänge des Fensters. Der letzte Abschnitt hat automatisch eine Länge von 1/4 der Gesamtfensterlänge.

SB_SETTEXT	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
Wert	WM_USER + 1		
Beschreibung	SB_SETTEXT ist eine Meldung, die einen Text in einem gegebenen Abschnitt der Statusbar schreibt.		
wParam&	Null-terminierte Angabe des Abschnitts, in dem der String stehen soll. 0 = erster Abschnitt, 1 = zweiter Abschnitt, usw.		
lParam%	Zeiger auf einen Null-terminierten String.		
Rückgabe:	BOOL: wenn erfolgreich, TRUE wird zurückgegeben, sonst FALSE		

Beispiel: Einfügen von Strings in eine zweigeteilte Statusbar (siehe Bild oben)

```
a$ = "Bereit" + CHR$(0)
~SendMessage (hwndStatusbar, SB_SETTEXT, 0, V:a$)

zeit$ = TIME$ + CHR$(0)
~SendMessage (hwndStatusbar, SB_SETTEXT, 1, V:zeit$)
```

Zu der CreateStatusbar()-Funktion mit seinen Meldungen, siehe auch das Beispiel zu STANDARD.GFW

Aufruf einer Statusbar-Leiste in einer Window Callback-Funktion:

```
CB () WIN (1) 0 TO -1 WindowProc (w,w,w,l)
```

```
PROCEDURE WindowProc (hwnd&, iMsg&, wParam&, lParam%)
```

```
SWITCH iMsg&
```

```
CASE WM_CREATE
```

```
/* Aufruf mit DLL-Funktion
```

```
ID_STATUSBAR = 100
```

```
SBARS_SIZEGRIP = $100
```

```
style% = WS_CHILD | WS_VISIBLE | SBARS_SIZEGRIP
```

```
hwndStatusbar& = ^CreateStatusbar (style%, "Bereit", hwnd, ID_STATUSBAR)
```

oder:

```
/* Aufruf mit normaler Windows-Funktion unter Verwendung der "statusbar_class"-Fensterklasse.
```

```
SBARS_SIZEGRIP = $100
```

```
style% = WS_CHILD | WS_VISIBLE | SBARS_SIZEGRIP
```

```
hwndStatusbar& = CreateWindow ("statusbar_class", "Bereit", style%, 0, 0, 0, 0, WIN (1), 100, _INSTANCE, 0)
```

```
RETVAL FALSE
```

Die Einbindung der Statusbar erfolgt in der WM_SIZE-Message des Hauptfensters.

Die Länge richtet sich nach der Angabe der Länge des Clientbereichs des Hauptfensters.

Die Höhe der Statusbar-Leiste ergibt sich automatisch aus dem gewählten FONT für Icon-Beschriftung.

```
CASE WM_SIZE
```

```
~GetClientRect (hwnd&, *rect.)
```

```
cxClient = rect.right
```

```
cyClient = rect.bottom
```

```
IF IsWindowVisible (hwndStatusbar&)
```

```
~SendMessage (hwndStatusbar&, WM_SIZE, 0, 0)
```

```
~GetWindowRect (hwndStatusbar&, *rect.)
```

```
cyStatus& = rect.bottom - rect.top
```

```
ELSE
```

```
cyStatus& = 0
```

```
ENDIF
```

```
cyClient = cyClient - cyStatus&
```

```
~MoveWindow (hwndClient&, 0, cyStatus&, cxClient&, cyClient&, TRUE)
```

```
~UpdateWindow (hwndClient&)
```

```
RETVAL FALSE
```

```
DEFAULT
```

```
RETVAL DefWindowProc (hwnd&, iMsg&, wParam&, lParam%)
```

```
ENDSWITCH
```

```
RETURN
```

Deshalb keine Parameter



WORD CreateToolBar (hwnd, style%, wID, nBitmaps, hBMInst, wBMID, lpButton%, nButtons, uStruktSize)

```

WORD  hwnd&           /* Fenster-Handle          */
LONG  style%          /* Fensterstile            */
WORD  wID&            /* Identifikations-Nummer  */
WORD  nBitmaps&       /* Anzahl Bitmaps          */
WORD  hBMInst         /* Resouce-Instance        */
WORD  hBMID           /* Resource-Indentifikation */
LONG  lpButton%       /* Zeiger auf Struktur     */
WORD  nButtons&       /* Anzahl Buttons          */
WORD  uStruktSize&    /* Strukturgröße           */

```

CreateToolBar() stellt ein Toolbar-Fenster unterhalb der Menüleiste dar und fügt spezielle Buttons hinzu. Die Toolbar ist mit dem Parent-Window verbunden. Die Applikation kann auch die CreateWindow()-Funktion mit der "toolbar_class"-Fensterklasse verwenden, muß aber über spezielle Toolbar-Messages die Buttons hinzufügen. Wie die Statusbar, so läuft diese Funktion auch unter Window NT. Gegenüber der Windowsfunktion *CreateToolBarEx()* ist diese Funktion auf das Wichtigste beschränkt worden.

Declaration:

```

DLL #1, "CTRLTOOL"
DECL WORD CreateToolBar (w,l,w,w,w,l,w,w)
ENDDLL

```

Parameter	Beschreibung	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
hwnd&	Parentfenster-Handle, zu dem die Toolbar gehört.			
style%	<p>WS_CHILD WS_VISIBLE ist vorgeschrieben.</p> <p>Folgender Toolbar-spezifische Stil kann verodert hinzugefügt werden, müssen aber zuvor initalisiert werden:</p> <p>CCS_NODIVIDER = \$40 Schaltet die Begrenzungslinie zur Menüleiste aus.</p> <p>TBSTYLE_TOOLTIPS = \$100 Ermöglicht den Aufruf eines Info-Fenster für jeden Button in der Toolbar (Quick-Info Fenster).</p>			
wID&	Identifikations-Nummer der Toolbar.			
nBitmaps&	Die Bitmaps der Buttons befinden sich in einem Streifen zusammengefaßt in dem Resourcen-File. Diese Variable steht für die gesamte Anzahl der einzelnen Bitmaps in diesem Streifen.			
hBMInst&	Instance des Programms (EXE, DLL), in der die Resource mit den Bitmaps eingebunden ist.			
hBMID&	Identifikation der Resource, in der die Bitmaps zu finden sind.			
lpButton%	Zeiger auf ein Array der TBBUTTON-Struktur, das die Informationen über die hinzuzufügenden Buttons enthält. Siehe dazu auch die Definition der TBBUTTON-Struktur weiter unten.			
nButtons&	Anzahl aller Buttons in der Toolbar, einschließlich aller SEPARATOR-Buttons.			
uStruktSize&	Größe des Arrays in Bytes.			
Rückgabe:	Bei Erfolg, das Handle der Toolbar, sonst NULL bei Mißerfolg.			

TBBUTTON Definition:

TYPE TBBUTTON:

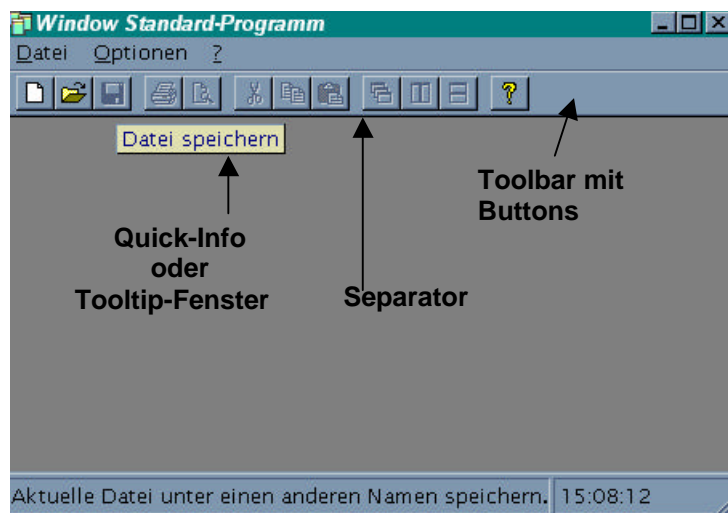
- **WORD** iBitmap
- **WORD** idCommand
- **BYTE** fsState
- **BYTE** fsStyle
- **LONG** dwData
- **WORD** iString

ENDTYPE

Variablentypen

iBitmap	Null-basierender Index des Button-Bitmaps. Index ist die Stelle, an der sich das Bitmap im Bitmapstreifen befindet.
idCommand	ID_Nummer des Buttons. Diese Identifikationsnummer wird in der WM_COMMAND-Message benutzt, wenn der Button gedrückt wird. Wenn der fsStyle-Stil TBSTYLE_SEP gesetzt ist, ist diese Variable Null.
fsState	Dieser Variablentyp kann aus folgenden Werten bestehen: TBSTATE_CHECKED = \$01 /* wird unterstützt */ TBSTATE_PRESSED = \$02 /* nicht unterstützt */ TBSTATE_ENABLED = \$04 /* wird unterstützt */ TBSTATE_HIDDEN = \$08 /* nicht unterstützt */
fsStyle	Dieser Variablentyp kann aus folgenden Werten bestehen: TBSTYLE_BUTTON = \$00 /* Button */ TBSTYLE_SEP = \$01 /* Separator */ TBSTYLE_CHECK = \$02 /* Check-Button
dwData	wird nicht benutzt. Nur aus Kompatibilitätsgründen zur COMMCTRL.DLL
iString	NULL-basierender Index für den Button-String. Jeder Button kann mit einem Hilfstext mittels TB_ADDSTRING-Message versehen werden. Wenn der fsStyle-Stil TBSTYLE_SEP gesetzt ist, ist diese Variable Null.

Zu der CreateToolbar()-Funktion mit seinen Meldungen, siehe das Beispiel zu STANDARD.GFW



Die Toolbar-Messages

TB_ADDBITMAP

WINDOWS 3.11

WINDOWS 95

WINDOWS NT

Wert **WM_USER + 19**

Beschreibung TB_ADDBITMAP ist eine Meldung, die eine oder mehrere Bitmaps zu den vorhandenen Buttons hinzufügt. Wird die Toolbar mittels CreateWindow()-Funktion aufgerufen, muß die Applikation die TB_BUTTONSTRUCTSIZE-Message aufrufen, bevor die TB_ADDBITMAP-Message gesendet wird.

wParam& Gesamtzahl der Bitmaps im Bitmapstreifen der Resource.

lParam% Zeiger auf eine TBADDBITMAP-Struktur. Beschreibung der Struktur etwas weiter unten.

Rückgabe: BOOL: wenn erfolgreich, TRUE wird zurückgegeben, sonst FALSE

TBADDBITMAP Definition:

TYPE TBADDBITMAP:

- **WORD** tblnst

- **WORD** tbnID

ENDTYPE

Variablentypen

tblnst Instance, in der die Resource mit dem Bitmapstreifen zu finden ist. Sollen die in der CTRLTOOL.DLL befindlichen Bitmaps verwendet werden, muß dieser Wert : HINST_COMMCTRL = -1 sein.

tbnID Identifikations-Nummer der Bitmap-Resourcen. Sollen die in der CTRLTOOL.DLL befindlichen Bitmaps verwendet werden, müssen folgende Werte hier angegeben werden:

IDB_STD_SMALL_COLOR = 0 wParam oder nBitmaps = 15



IDB_STD_LARGE_COLOR = 1 wParam oder nBitmaps = 15



IDB_VIEW_SMALL_COLOR = 4 wParam oder nBitmaps = 12



IDB_VIEW_LARGE_COLOR = 5 wParam oder nBitmaps = 12



Wird tblnst gleich NULL gesetzt, muß die Variable tbnID das Handle eines Bitmaps in einer Resource sein..

Wert **WM_USER + 20**

Beschreibung TB_ADDBUTTONS ist eine Meldung, die eine oder mehrere Buttons in eine vorhandene Toolbar einfügt. Wird die Toolbar mittels der CreateWindow()-Funktion erzeugt, muß zuvor die TB_BUTTONSTRUCTSIZE, dann die TB_ADDBITMAP-MESSAGE gesendet werden, bevor diese Meldung abgeschickt werden kann.

wParam& Anzahl der Buttons, die hinzugefügt werden sollen.

lParam% Zeiger auf ein Array der TBBUTTON-Struktur, die die Informationen über die hinzuzufügenden Buttons enthält. Siehe dazu auch die Definition der TBBUTTON-Struktur.

Rückgabe: BOOL: wenn erfolgreich, TRUE wird zurückgegeben, sonst FALSE

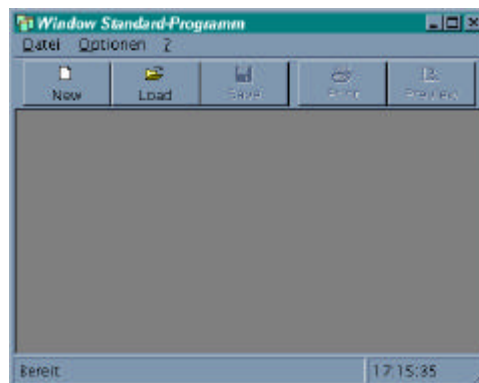
Wert **WM_USER + 28**

Beschreibung TB_ADDSTRING ist eine Meldung, die einen Hilfs-String unterhalb der Bitmaps der Toolbar-Buttons schreibt.

wParam& Wird auf Null gesetzt.

lParam% Zeiger auf einen String, der aus mehreren Teilstrings, getrennt mit CHR\$(0) besteht. Siehe dazu die Beschreibung der CTRLTOOL.DLL-Funktion *GetString()*.

Rückgabe: Keine.

**Wert** **WM_USER + 33**

Beschreibung TB_AUTOSIZE ist eine Meldung, die eine Toolbar neu in ihrer Größe darstellt. Sie wird immer dann gesendet, wenn das Hauptfenster eine WM_SIZE-MESSAGE verarbeitet, oder wenn neue Buttons der Toolbar hinzugefügt werden.

wParam& Wird auf 0 gesetzt.

lParam% Wird auf 0 gesetzt.

Rückgabe: Keine

TB_BUTTONSTRUCTSIZE**WINDOWS 3.11****WINDOWS 95****WINDOWS NT****Wert** **WM_USER + 30**

Beschreibung TB_BUTTONSTRUCTSIZE ist eine Meldung, die die Größe der TBBUTTON-Struktur enthält. Wenn die Application die CreateWindow()-Funktion verwendet, muß diese Meldung zuerst gesendet werden, bevor die TB_ADDBITMAP oder die TB_ADDBUTTONS-Message abgeschickt wird. Die CreateToolbar()-Funktion dieser DLL ruft intern diese Meldung automatisch auf.

wParam& Die Größe in Bytes der TBBUTTON-Struktur. Siehe die Definition der TBBUTTON-Struktur unter der CreateToolbar()-Funktion.

lParam% Wird auf 0 gesetzt.

Rückgabe: Keine.

TB_COMMANDSTRINGTOINDEX**WINDOWS 3.11****WINDOWS 95****WINDOWS NT****Wert** **WM_USER + 25**

Beschreibung TB_COMMANDTOINDEX ist eine Meldung, die den Index eines Buttons zurück liefert, wenn die Identifikationsnummer angegeben wird.

wParam& ID-Nummer des Buttons.
(z.B.: IDM_NEW = 100, IDM_OPEN = 101,...)

lParam% Wird auf 0 gesetzt.

Rückgabe: Null-basierender Index des Buttons. Index: durchlaufende Nummer der Buttons einschließlich der Separatoren von Null beginnend.

TB_CHECKBUTTON**WINDOWS 3.11****WINDOWS 95****WINDOWS NT****Wert** **WM_USER + 2**

Beschreibung TB_CHECKBUTTON ist eine Meldung, die einen vorhandenen Button als gedrückt oder nicht gedrückt darstellt.

wParam& Kommando-Identifikationsnummer des Buttons.

lParam% **TRUE:** Button ist nicht gesperrt. **FALSE:** Button ist gesperrt.

Rückgabe: BOOL: wenn erfolgreich, TRUE wird zurückgegeben, sonst FALSE

TB_ENABLEBUTTON**WINDOWS 3.11****WINDOWS 95****WINDOWS NT****Wert** **WM_USER + 1**

Beschreibung TB_ENABLEBUTTON ist eine Meldung, die einen vorhandenen Button sperrt oder in Funktion setzt.

wParam& Kommando-Identifikationsnummer des Buttons.

lParam% **TRUE:** Button ist nicht gesperrt.
FALSE: Button ist gesperrt.

Rückgabe: BOOL: wenn erfolgreich, TRUE wird zurückgegeben, sonst FALSE

TB_GETITEMRECT**WINDOWS 3.11****WINDOWS 95****WINDOWS NT****Wert** **WM_USER + 29**

Beschreibung TB_GETITEMRECT ist eine Meldung, die die Rechteckkoordinaten eines vorhandenen Buttons zurückliefert.

wParam& Null-basierender Index des Buttons, dessen Koordinaten innerhalb der Toolbar erfragt werden soll.

lParam% Zeiger auf eine Rechteck-Datenstruktur, in der die Koordinaten geschrieben werden.

Rechteck-Datenstruktur:

TYPE RECT:
- **WORD** left
- **WORD** top
- **WORD** right
- **WORD** bottom
ENDTYPE

Rückgabe: BOOL: wenn erfolgreich, TRUE wird zurückgegeben, sonst FALSE

TB_GETSTATE**WINDOWS 3.11****WINDOWS 95****WINDOWS NT****Wert** **WM_USER + 18**

Beschreibung TB_GETSTATE ist eine Meldung, die den Status eines Button zurückliefert.

wParam& Kommando-Identifikationsnummer des Buttons, für den der Status zurück geliefert werden soll.

lParam% Wird auf 0 gesetzt.

Rückgabe: Wenn erfolgreich, wird der Button-Status zurückgeliefert, sonst 1 bei Mißerfolg. Der Rückgabewert kann eine Kombination aus folgenden Werten sein.

TBSTATE_CHECKED wird unterstützt
TBSTATE_ENABLED wird unterstützt

TB_GETTOOLTIPS**WINDOWS 3.11****WINDOWS 95****WINDOWS NT****Wert** **WM_USER + 35**

Beschreibung TB_GETTOOLTIP ist eine Meldung, die das Handle eines ToolTip zurückliefert, wenn eine vorhanden ist. Daß heißt, wenn die Toolbar mit dem TBSTYLE-TOOLTIP versehen worden ist.

wParam& Wird auf 0 gesetzt.

lParam% Wird auf 0 gesetzt.

Rückgabe: Das Handle des ToolTip-Controls, das zur Toolbar gehört.
NULL wird zurück geliefert, wenn die Toolbar kein ToolTip besitzt.

Ein Tooltip-Control ist die Bezeichnung für das Quick-Info Fenster, das bei Berührung des Buttons mit dem Cursor unterhalb des Cursors erscheint..

TB_ISBUTTONCHECKED**WINDOWS 3.11****WINDOWS 95****WINDOWS NT****Wert** **WM_USER + 10**

Beschreibung TB_ISBUTTONCHECKED liefert den Zustand eines Toolbar-Buttons zurück, der zuvor mit dem TBSTYLE_CHECK Stil definiert wurde.

wParam& Kommando-Identifikationsnummer des Buttons.

lParam% Wird auf 0 gesetzt.

Rückgabe: 1, wenn Button ist checked. 0, wenn nicht.

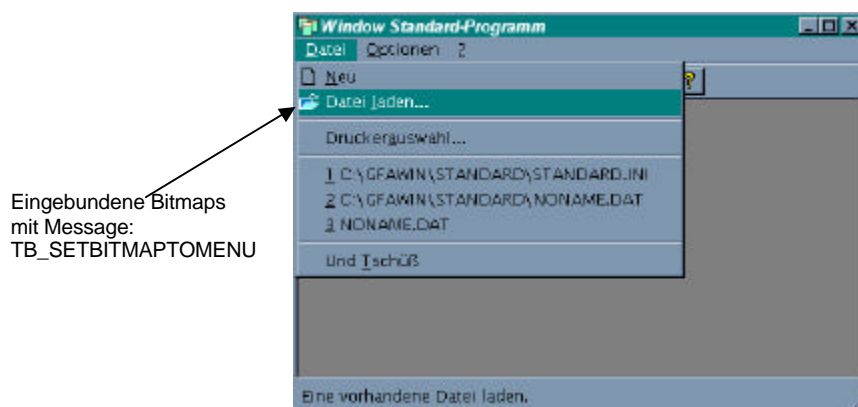
TB_SETBITMAPTOMENU**WINDOWS 3.11****WINDOWS 95****WINDOWS NT****Wert** **WM_USER + 36**

Beschreibung TB_SETBITMAPTOMENU ist eine neue Meldung, die in der originalen COMMCTRL.DLL nicht vorhanden ist. Diese Meldung setzt links neben dem Menueintrag, wo sich sonst das Häkchen bei dem Menüstil MF_CHECKED befindet, das Bitmap des Buttons, das denselben Befehl aufruft. Dies funktioniert nur, wenn Menüeintrag und Toolbar-Button dieselbe Kommando-Identifikationsnummer besitzen.

wParam& Handle des Menüs eines Hauptfensters, das mit der Funktion GetMenu() ermittelt werden kann. Die Meldung bewirkt, das das ganze Menu, Menüpunkt nach Menüpunkt, durchsucht wird. Wenn gleiche Identifikationsnummern gefunden werden, wird das entsprechend Bitmap in das Menu eingebunden

lParam% **TRUE:** Einschalten der Funktion.
FALSE: Ausschalten der Funktion.

Rückgabe: Keine.



Die folgenden Messages sind aus der Window 95-API übernommen worden. Sie müssen vor dem Aufruf mit den angegebenen Werten initialisiert werden. Diese Meldungen werden in der Parent-Window CallBack-Funktion abgefragt, wenn eine Toolbar mit dem Toolbar-Stil TBSTYLE_TOOLTIPS (Tooltips erlaubt) im Programm eingebunden wird.

WM_NOTIFY	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
Wert	\$004E		
Beschreibung	WM_NOTIFY ist eine Meldung, die das Parent-Window eines ToolbarControls drüber informiert, das ein Ereignis sich im Control abspielt oder das das Control einige Informationen benötigt.		
wParam&	Handle des Controls einer Toolbar, das die Meldung schickt.		
lParam%	Zeiger auf eine NMHDR-Struktur, die den Notifikation-Code und noch einige andere Informationen enthält.		
	TYPE NMHDR: - LONG hwnFrom - LONG idFrom - LONG code ENDTYPE		
Typen-Variable			
hwnFrom	Das Window-Handle des Controls, das die Meldung sendet.		
idFrom	Die ID-Nummer des Controls, das die Meldung sendet.		
code	Der Norifikation-Code eines Controls.		
	Es können folgende Tooltip-Notifikation Meldungen auftreten:		
	TTN_NEEDTEXT	\$FDF8	
	TTN_SHOW	\$FDF7	
	TTN_POP	\$FDF6	
Rückgabe:	Der Wert wird ignoriert.		

Wert \$FDF8

Beschreibung TTN_NEEDTEXT ist eine Tooltip-Notifikation-Meldung, die sich einen Text für ein Tooltip holt. Diese Notifikation-Meldung wird immer dann an ein Fenster gesendet, wenn sich ein Ereignis in einem Control abspielt und der Toolbar-Stil TBSTYLE_TOOLTIPS einer Toolbar gesetzt ist. Diese Meldung wird in Form einer WM_NOTIFY-Meldung gesendet.

wParam& ID-Nummer eines Toolbar-Controls.

lParam% Zeiger auf eine TOOLTIPTEXT-Struktur.

So ist es in der Windows 95-API geregelt:

TYPE TOOLTIPTEXT:

NMHDR	nmhdr	Eingeschlossene NMHDR-Struktur
- LONG	lpszText	Zeiger auf einen Text
- CHAR*80	szText\$	Der Text selber
- WORD	hinst	Instance
- LONG	uFlags	Flag
ENDTYPE		

In GFA-Basic ist es nicht möglich, eine Struktur in einer Struktur darzustellen. Deswegen ist dieser Klimmzug mit den Dummy Type-Variablen nmhdr1-3 nötig.

TYPE TOOLTIPTEXT:

- LONG	nmhdr1
- LONG	nmhdr2
- LONG	nmhdr3
- LONG	lpszText
- CHAR*80	szText\$
- WORD	hinst
- LONG	uFlags
ENDTYPE	

Typen-Variable

nmhdr1-3 Siehe die Definition der NMHDR-Struktur unter der WM_NOTIFY-Meldung.

lpszText Zeiger auf einen String, der den Text für ein Tooltip enthält.
Ist hinst ein Handle einer Instance, muß diese Variable eine Identifikation auf einen Ressourcen-String sein.

szText\$ Ein Puffer, der den Tooltip-Text enthält. Die Applikation kann anstelle einer String-Resource einen Text in diesen Puffer schreiben.

hinst Das Instance-Handle einer String-Resource, das für den Tooltip-Text benötigt wird. Ist lpszText ein Zeiger auf einen Tooltip-Text, ist diese Variable NULL.

uFlags Wird in dieser Version der CTRLTOOL.DLL nicht unterstützt.

Rückgabe: Keine.

TTN_SHOW	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
-----------------	---------------------	-------------------	-------------------

Wert	\$FDF7
Beschreibung	TTN_SHOW ist eine Tooltip-Notifikation-Meldung, die dem Parent-Window mitteilt, daß das Tooltip sichtbar ist. Diese Notifikation-Meldung wird in Form einer WM_NOTIFY-Meldung gesendet.
wParam&	Die ID-Nummer des Tooltip-Controls.
lParam%	Zeiger auf eine NMHDR-Struktur, die Informationen über die Notifikation-Meldung enthält. Siehe dazu die Beschreibung der NMHDR-Struktur unter der WM_NOTIFY-Meldung.
Rückgabe:	Keine.

TTN_POP	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
----------------	---------------------	-------------------	-------------------

Wert	\$FDF6
Beschreibung	TTN_POP ist eine Tooltip-Notifikation-Meldung, die dem Parent-Window mitteilt, daß das Tooltip nicht mehr sichtbar ist. Diese Notifikation-Meldung wird in Form einer WM_NOTIFY-Meldung gesendet.
wParam&	Die ID-Nummer des Tooltip-Controls.
lParam%	Zeiger auf eine NMHDR-Struktur, die Informationen über die Notifikation-Meldung enthält. Siehe dazu die Beschreibung der NMHDR-Struktur unter der WM_NOTIFY-Meldung.
Rückgabe:	Keine.

Die Tooltip-Meldung

TTM_ADDTOOL	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
--------------------	---------------------	-------------------	-------------------

Wert	WM_USER + 4
Beschreibung	TTM_ADDTOOL ist eine Tooltip-Meldung, die für ein eigenes Control (z.B.: Combobox) ein Tooltip-Fenster erstellt.
wParam&	Wird auf Null gesetzt.
lParam%	Zeiger auf eine TOOLINFO-Struktur, die die Information enthält, die das Tooltip Control zur Anzeige des Tooltip-Textes benötigt.
Rückgabe:	BOOL: wenn erfolgreich, TRUE wird zurückgegeben, sonst FALSE

TOOLINFO-Definition

TYPE TOOLINFO:

```
- USHORT    cbSize           /* in commctrl.h als UINT
- USHORT    cbuFlags        /* in commctrl.h als UINT
- WORD      cbhwnd
- USHORT    cbuld           /* in commctrl.h als UINT
- LONG      cbxy            /* 1/2 RECT-Datenstruktur (x und y)
- LONG      cblh            /* 1/2 RECT-Datenstruktur (l und h)
- WORD      cbinst         /* Instance
- LONG      szcbText        /* Zeiger auf Textpuffer
ENDTYPE
```

Typen-Variable

cbSize	Die Größe in Bytes der TTMINFO-Struktur. Diese Variable muß zuerst initialisiert werden, bevor die Struktur in der TTM_ADDTOOL-Meldung verwendet wird.		
cbuFlags	Diese Variable kann aus der Kombination der folgenden Werte bestehen, verbunden mit dem OR-Operator ().		
	TTF_CENTERTIP	\$02	Zentriert das Tooltip-Fenster mittig unter das Control.
	TTF_IDISHWND	\$01	Die cbuld-Variable ist das Window-Handle des eingesetzten neuen Controls in der Toolbar. Wenn nicht gesetzt, muß cbuld die ID-Nummer des Controls sein.
cbhwnd	Das Handle des Fensters, das das Control enthält.		
cbuld	Die applikationsdefinierte ID-Nummer des Controls. Wenn cbuFlags den TTF_IDISWND-Wert enthält, muß diese Variable das Handle des Controls sein.		
cbxy	Rect-Datenstruktur des Controls in der Toolbar.		
cbllh	Wird in dieser Version intern in der CTRLTOOL.DLL benötigt.		
cbinst	Das Instance-Handle des Moduls, das die String-Resource für das Control enthält. Wenn szcbText der ID-Wert der String-Resource ist, wird diese Variable benötigt.		
szcbText	Zeiger auf einen Puffer, der den Text für das Control oder die ID-Nummer der String-Resource enthält. Wenn diese Variable auf LPSTR_TEXTCALLBACK (-1) gesetzt wird, wird das Control eine TTN_NEEDTEXT-Meldung an das Parent-Window schicken, um den Text zu holen.		

Beispiel:

```

hwndTT = SendMessage( hwndToolBar, TB_GETTOOLTIPS, 0, 0)

DIM TOOLINFO: toolinfo.
toolinfo.cbSize = LEN(toolinfo.)
toolinfo.cbuFlags = TTF_CENTERTIP | TTF_IDISHWND oder toolinfo.cbuFlags = TTF_CENTERTIP
toolinfo.cbhwnd = hwndToolBar                                toolinfo.cbhwnd = hwndToolBar
toolinfo.cbuld = hwndCombo                                  toolinfo.cbuld = ID_COMBO

1.)
toolinfo.cbinst = _INSTANCE
toolinfo.szcbText = ID_COMBO

2.)
toolinfo.cbinst = 0
a$ = „1, 2, 3 ComboBox“ + CHR$(0)
toolinfo.szcbText = V:a$

3.)
toolinfo.cbinst = 0
toolinfo.szcbText = LPSTR_TEXTCALLBACK

~SendMessage( hwndTT, TTM_ADDTOOL, 0, *toolinfo.)

```

BOOL MenuHelp (hwnd, hInst, maxMenu, wParam, lParam, lp%)

WORD hwnd& /* Statusbar-Handle */
WORD hInst& /* Instance */
WORD wParam& /* 16-Bit Message */
LONG lParam% /* 32-Bit-Message */
LONG lp% /* Zeiger auf Array */

Die **MenuHelp()**-Funktion stellt einen Hilfe-Text für ein angewähltes Menu-Item in einer vorhandenen Statusbar dar. Die Funktion wird in Zusammenhang mit der WM_MENUSELECT-Message des Haupt-Fensters benutzt, um ein Hilfe-Text darzustellen, wenn ein Menüpunkt angewählt worden ist.

Diese Funktion kann nur verwendet werden, wenn das Menu mit Hilfe eines Ressourcen Workshops aufgebaut wird, und zusammen mit dem GFA-Basic Programm unter Verwendung des Compilers compiliert wird.

Declaration:

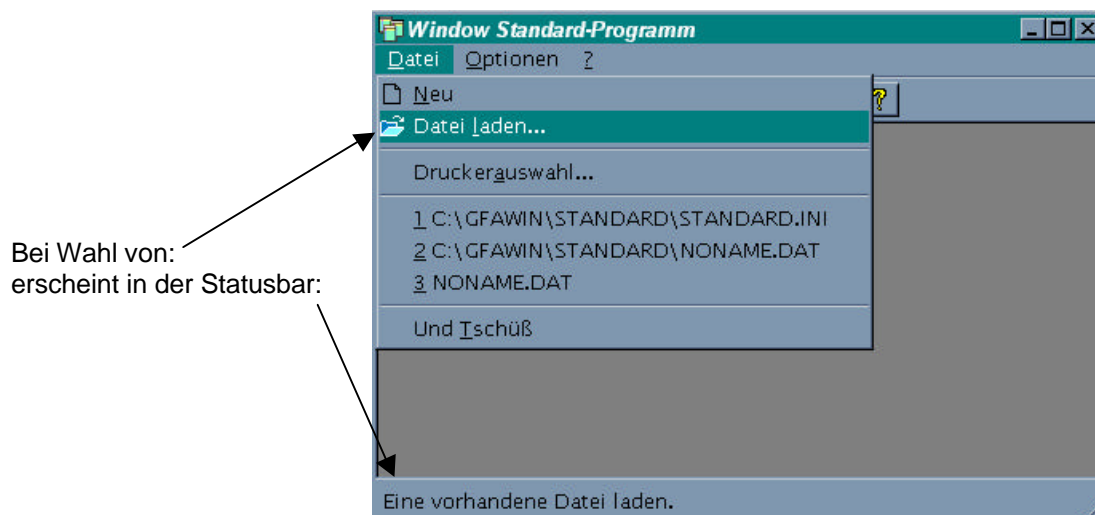
```
DLL #1, "CTRLTOOL"  
DECL BOOL MenuHelp (w,w,w,w,l,l)  
ENDDLL
```

Parameter	Beschreibung	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
hwnd&	Handle des Statusbar-Fensters (z.B.: hwndStatusbar)			
hInst&	Instance des Programms, in dem das Menu eingebunden ist.			
maxMenu&	Maximale Anzahl der Menü-Punkte in dem Menü. (z.B.: Datei, Bearbeiten, Ansicht,...)			
wParam&	16-Bit Messageteil der WM_MENUSELECT-Meldung.			
lParam%	32-Bit Messageteil der WM_MENUSELECT-Meldung.			
lp%	Zeiger auf ein LONG-Array, das ein Paar aus "String Resource ID-Nummern" und Menu-Handle enthält. Die Funktion sucht das Array nach dem Handle des gewählten Menüs durch und, wenn gefunden, lädt es den dazugehörigen String aus der Resource.			

Rückgabe:

BOOL: wenn erfolgreich, TRUE wird zurückgegeben, sonst FALSE

Zu der MenuHelp()-Funktion, siehe das Beispiel zu STANDARD.GFW



BOOL SendSpinMessage (hwnd, style&, lpsz1%, lpsz2%, wVon, wBis, Vfarbe%, Hfarbe%)

```

WORD  hwnd&      /* Handle des Dialogelements */
WORD  style&      /* SpinButton Stil          */
LONG  lpsz1%      /* Adresse eines Strings     */
LONG  lpsz2%      /* Adresse eines Strings     */
WORD  wVon&       /* Integer-Zahl             */
WORD  wBis&       /* Integer-Zahl             */
LONG  Vfarbe%     /* RGB-Wert                 */
LONG  Hfarbe%     /* RGB-Wert                 */

```

Die SendSpinMessage()-Funktion sendet eine Initialisierungs-Nachricht an ein Dialogfenster das zuvor in der übergeordneten Dialogbox mit der "spin_class"-Klasse deklariert wurde.

Es wird ein Editierfenster mit Up/Down-Buttons dargestellt.

Wird in der eigenen Applikation keine Dialogboxen verwendet, entfällt der Aufruf von DLG 3D ON/OFF.

Declaration:

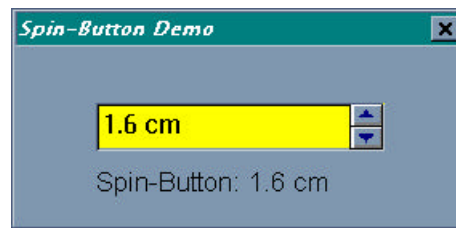
```

DLL #1, "CTRLTOOL"
DECL BOOL SendSpinMessage (w,w,l,l,w,w,l,l)
ENDDLL

```

Parameter	Beschreibung	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
hwnd&	Handle des Fensters.			
style&	SpinButton-Stil mit:	SPS_RIGHT = \$00	Up/Down Buttons rechts vom Editierfenster.	
		SPS_LEFT = \$01	Up/Down Button links vom Editierfenster.	
lpsz1%	NULL-terminierter String, der den aktuellen Eintrag (Zahl) im Editierfenster beim ersten Aufruf der Dialogbox anzeigt.			
lpsz2%	NULL-terminierter String, der die Schrittfolge sowie die entsprechende Größenangabe enthält. Beispiel: "1 cm" oder "0.1 mm". Wird keine Größenangabe angegeben, muß nach der Zahl eine Leerstelle (x) folgen. Beispiel: „1x„.			
wVon&	Untere Zählgrenze.			
wBis&	Obere Zählgrenze.			
Vfarbe%	Schriftfarbe des Editierfensters als RGB-Wert. Wird 0 angegeben, wird die System-Textfarbe verwendet.			
Hfarbe%	Hintergrundfarbe des Editierfensters als RGB-Wert. Wird 0 angegeben, wird die System-Fensterhintergrundfarbe verwendet.			
Rückgabe	immer TRUE (1).			
Ergebnis	Auslesen des Editfensters in der WM_COMMAND-Message.			
ID-Nr	wParam a\$ = _WIN\$ (DLGITEM (1, _wParam))			
hwnd	LOWORD (_IParam) = Handle des Spin-Buttons.			
SPN_CHANGE	HIWORD (_IParam) = 1, wenn eine Änderung im Editfenster stattfindet.			

Beispiel: Einbindung der ^SendMessage()-Funktion in eine Dialogbox.



```
DLL #1,"CTRLTOOL.DLL"
DECL BOOL InitCtrlTool()
DECL BOOL SendMessage (w,w,l,l,w,w,l,l)
ENDDLL

~^InitCtrlTool()
,
DLG 3D ON /* 3D-Darstellung für die Dialogbox
,
style% = DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
DIALOG #0, 70, 5, 285, 138, "Spin-Button Demo", style%, -17, "Arial"
CONTROL "", 100, "spin_class", WS_CHILD | WS_VISIBLE, 50, 35, 180, 30
CONTROL "Spin-Button: ", 101, "Static", WS_CHILD | WS_VISIBLE | SS_LEFT, 50, 75, 180, 30
ENDDIALOG
SHOWDIALOG #0
,
SPS_RIGHT = $00
~^SendMessage (DLGITEM (0, 100), SPS_RIGHT, "1", "0.1 cm", 0, 10, RGB (0, 0, 0), RGB (255, 255, 0))
,
DO
GETEVENT
SELECT _Mess
CASE WM_COMMAND
SELECT _wParam
CASE 100
IF HIWORD (_iParam) = SPN_CHANGE /* SPN_CHANGE = 1
a$ = "Spin-Button: " + _WIN$ (DLGITEM (0, 100)) + CHR$(0)
~SendMessage (DLGITEM (0, 101), WM_SETTEXT, 0, V:a$)
ENDIF
ENDSELECT
ENDSELECT
UNTIL MENU(1) = 4
,
CLOSEDIALOG #0
DLG 3D OFF
FREEDLL 1
END
```

Sie auch das Beispiel zur ^SendKarteiMessage()-Funktion.

BOOL SendKarteiMessage (hwnd, wNr, wVon, wBis, cColor%)

WORD hwnd& /* Handle des Dialogelements */
WORD wNr& /* Karteikartennummer */
WORD wVon& /* Integer-Zahl */
WORD wBis& /* Integer-Zahl */
LONG cColor% /* RGB-Wert */

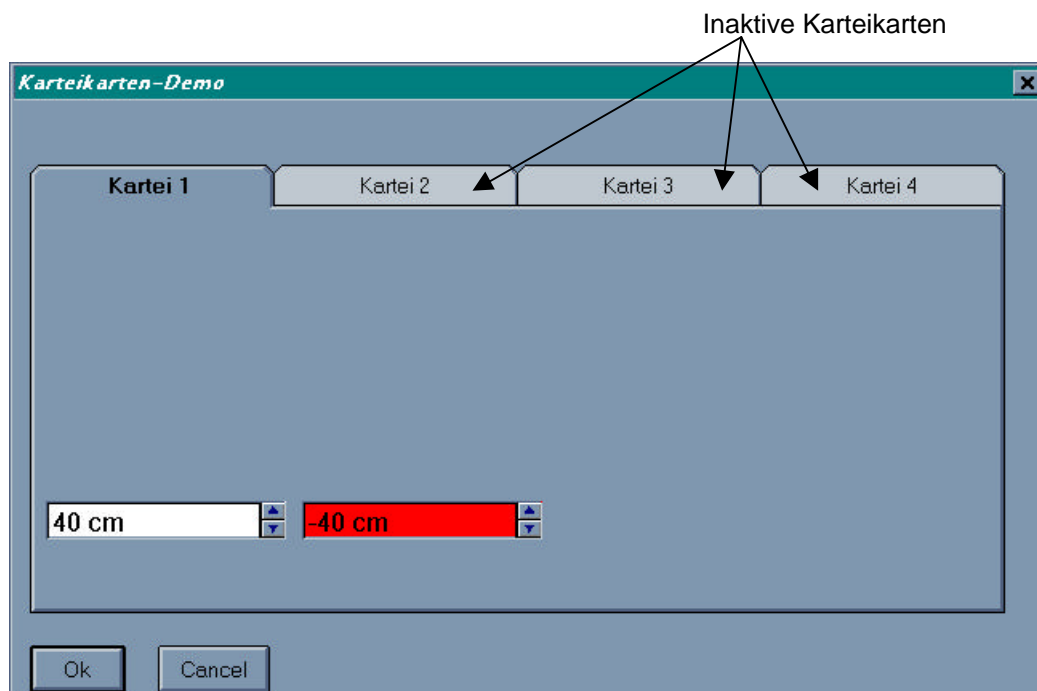
Die SendKarteiMessage()-Funktion wird an ein Dialogelement geschickt, das in der übergeordneten Dialogbox mit dem "Kartei"-Klasse deklariert werden muß. Für jede Karteikarte muß diese Funktion aufgerufen werden.

Wird in der eigenen Applikation keine Dialogboxen verwendet, entfällt der Aufruf von DLG 3D ON/OFF.

Declaration:

```
DLL #1, "CTRLTOOL"  
DECL BOOL SendKarteiMessage (w,w,w,w,l)  
ENDDLL
```

Parameter	Beschreibung	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
hwnd&	Handle des Fensters.			
wNr&	Fortlaufende Nummer der Karteikarten. Anzahl der Karteikarten wird im Fenstertext des Dialogelements mittels dem Titeltex, getrennt mit „," festgelegt.			
wVon&	Erstes Dialogelement, das zu dieser Karteikarte gehört.			
Wbis&	Letztes Dialogelement, das zu dieser Karteikarte gehört.			
Ccolor%	Farbe der inaktiven Karteikarte. Wird 0 angegeben, wird die inaktive Karteikarte auf hellgrau gesetzt.			
Rückgabe:	Immer TRUE (1).			



Beispiel: Einbindung der Karteikarte in eine Dialogbox.

```
DEFWRD "a-z"
OPENW #1, 10, 10, 600, 400, -1
'
DLL #1,"CTRLTOOL.DLL"
  DECL BOOL InitCtrlTool()
  DECL BOOL SendSpinMessage(w,w,l,l,w,w,l,l)
  DECL BOOL SendKarteiMessage(w,w,w,w,l)
ENDDLL
'
~^InitCtrlTool()
SPN_CHANGE = 1, SPS_RIGHT = $00
NULL$ = CHR$(0)
DLG 3D ON
'
style% = DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
DIALOG #2, 300, 300, 650, 400, "Karteikarten-Demo", style%, -12, "Ms Sans Serif"
  CB DLG(2) 0 TO -1 CBX()
  CONTROL "", 10, "spin_class", WS_TABSTOP, 20, 220, 150, 24
  CONTROL "0",11,"spin_class", WS_TABSTOP, 180, 220, 150, 24
  CONTROL "",20,"spin_class", WS_TABSTOP, 20, 250, 150, 24
  CONTROL "0",21,"spin_class", WS_TABSTOP, 180, 250, 150, 24
  DEFPUSHBUTTON "&Ok", IDOK, 10, 340, 60, 30, WS_TABSTOP | BS_DEFPUSHBUTTON
  PUSHBUTTON "&Cancel", IDCANCEL, 90, 340, 70, 30, WS_TABSTOP
' Das Control "Kartei" muß immer an letzter Stelle der Dialogboxstruktur stehen!!
  CONTROL "Kartei 1;Kartei 2;Kartei 3;Kartei 4", 400, "kartei_class", WS_TABSTOP, 10, 40, 610, 280
ENDDIALOG
'
~^SendKarteiMessage (DLGITEM( 2, 400), 1, 20, 21, 0)
~^SendKarteiMessage (DLGITEM( 2, 400), 2, 10, 11, 0)
~^SendKarteiMessage (DLGITEM( 2, 400), 3, 20, 21, 0)
'
a$ = "50" + NULL$
b$ = "1 cm" + NULL$
~^SendSpinMessage (DLGITEM( 2, 10), SPS_RIGHT, V:a$, V:b$, -100, 100, RGB( 255, 255, 255), RGB( 0, 0, 255))
a$ = "-50" + NULL$
~^SendSpinMessage (DLGITEM( 2, 11), SPS_RIGHT, V:a$, V:b$, -100, 100, RGB( 0, 0, 0), RGB( 0, 255, 255))
a$ = "40" + NULL$
~^SendSpinMessage (DLGITEM( 2, 20), SPS_RIGHT, V:a$, V:b$, -100, 100, RGB( 0, 0, 0), RGB( 255, 255, 255))
a$ = "-40" + NULL$
~^SendSpinMessage (DLGITEM( 2, 21), SPS_RIGHT, V:a$, V:b$, -100, 100, RGB( 0, 0, 0), RGB( 255, 0, 0))
SHOWDIALOG #2
~SetFocus (DLGITEM (2, IDOK))
'
DO
  SLEEP
LOOP UNTIL antw = IDOK
CLOSEDIALOG #2
DLG 3D OFF
FREEDLL 1
CLOSEW #1
'
PROCEDURE CBX( hwnd, MSG, wParam, lParam%)
  SELECT MSG
  CASE WM_COMMAND
    SWITCH wParam
    CASE IDOK, IDCANCEL
      antw = IDOK
    CASE 10, 20
      IF HIWORD (lParam%) = SPN_CHANGE          /* SPN_CHANGE = 1
        puffer$ = _WIN$ (LOWORD (lParam%))
        TEXT 5, 10, "Control " + STR$ (wParam) + ": " + puffer$
      ENDIF
    CASE 11, 21
      IF HIWORD (lParam%) = SPN_CHANGE
        puffer$ = _WIN$ (LOWORD (lParam%))
        TEXT 5, .30, "Control " + STR$ (wParam) + ": " + puffer$
      ENDIF
    END SWITCH
  END SELECT
RETURN
```


Progress Bar (Fortschrittsanzeige)

Vor Aufruf eines Fortschrittsfensters muß unbedingt die InitCtrlTool()-Funktion aufgerufen werden. In ihr werden alle Fenster-Klassen der DLL initialisiert. Wird in der eigenen Applikation keine Dialogboxen verwendet, entfällt der Aufruf von DLG 3D ON/OFF.

PBM_DELTAPOS		WINDOWS 3.11	WINDOWS 95	WINDOWS NT
Wert	WM_USER + 3			
Beschreibung	PBM_DELTAPOS ist eine Meldung, die die aktuelle Balkenposition in einer Progress Bar um den inkrementellen Betrag bewegt und die Anzeige der Progress Bar aktualisiert.			
wParam&	Der Betrag, um den die derzeitige Position der Balkens erhöht wird.			
lParam%	Wird auf Null gesetzt.			
Rückgabe:	Die vorherige Position des Balkens wird zurückgegeben.			

PBM_SETCOLOR		WINDOWS 3.11	WINDOWS 95	WINDOWS NT
Wert	WM_USER			
Beschreibung	PBM_SETCOLOR ist eine Meldung, mit der die Balkenfarbe einer Progress Bar gesetzt wird. Diese Meldung ist neu und ist nicht Bestandteil der WIN95API.			
wParam&	Normal ist Null. Wird 1 angegeben, so wird der Balken mit der Systemfarbe 'COLOR_HIGHLIGHT' eines angewählten Items dargestellt. Dies ist für das Zurücksetzen einer vorherigen gewählten Farbe gedacht. Dabei muß lParam% gleich Null sein.			
lParam%	RGB-Wert, der die Balken-Farbe definiert. Voreinstellung ist diese Systemfarbe.			
Rückgabe:	Keine.			

PBM_SETPOS		WINDOWS 3.11	WINDOWS 95	WINDOWS NT
Wert	WM_USER + 2			
Beschreibung	PBM_SETPOS ist eine Meldung, die die aktuelle Balkenposition einer Progress Bar setzt und die Anzeige der Progress Bar aktualisiert.			
wParam&	Die neue Position des Balkens der Progress Bar			
lParam%	Wird auf Null gesetzt.			
Rückgabe:	Die vorherige Position wird zurückgegeben.			

PBM_SETRANGE**WINDOWS 3.11****WINDOWS 95****WINDOWS NT****Wert** **WM_USER + 1**

Beschreibung PBM_SETRANGE ist eine Meldung, die den Bereich (Range) über die Minimum- und Maximumwerte einer Progress Bar setzt. Es wird die Anzeige der Progress Bar aktualisiert.

wParam& Wird auf Null gesetzt.

lParam% LOWORD (lParam%)
Im niederwertigen Wort wird die untere Grenze angegeben.
Voreinstellung ist 0.
HIWORD (lParam%)
Im höherwertigen Wort wird die obere Grenze angegeben.
Voreinstellung ist 100.

Rückgabe: Keine.

PBM_SETSTEP**WINDOWS 3.11****WINDOWS 95****WINDOWS NT****Wert** **WM_USER + 4**

Beschreibung PBM_SETSTEP ist eine Meldung, die den Wert für das Inkrement der Progress Bar definiert. Dieses Inkrement bestimmt die Verlängerung des Balkens bei einer PBM_STEPIT-Meldung.

wParam& Der neue Wert für die Schrittweite des Balkens der Progress Bar.
Voreinstellung ist 10.

lParam% Wird auf Null gesetzt.

Rückgabe: Vorherige Schrittweite.

PBM_STEPIT**WINDOWS 3.11****WINDOWS 95****WINDOWS NT****Wert** **WM_USER + 5**

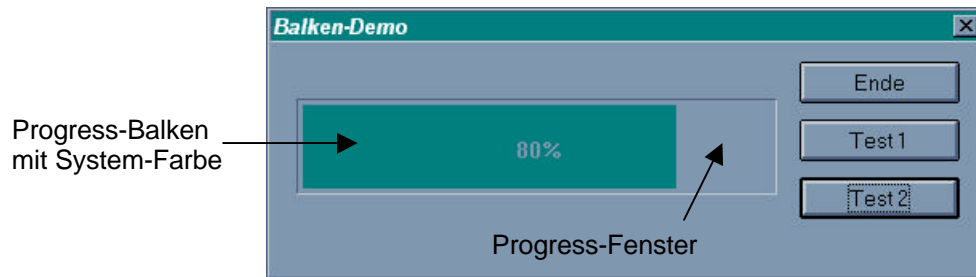
Beschreibung PBM_STEPIT ist eine Meldung, die die derzeitige Position des Balkens einer Progress Bar um die Schrittweite erhöht. Die Progress Bar wird neu gezeichnet, um die neue Position anzuzeigen. Die PBM_SETSTEP-Meldung wird benutzt, um die Schrittweite des Balkens festzulegen.
Wird der Maximum-Wert erreicht, startet der Balken wieder von dem eingestellten unteren Grenzwert (Anfangswert).

wParam& Wird nicht benutzt.

lParam% Wird nicht benutzt.

Rückgabe: Vorherige Position der Progress Bar.

Beispiel: Dialogbox mit einer Progress Bar:



Quelltext:

```
$LNK EXE C:\GFAWIN\PROGRESS.EXE
```

```
@WinMain()
```

```
,
```

```
PROCEDURE WinMain()
```

```
DEFWRD "a-z"
```

```
DLL #1,"CTRLTOOL.DLL"
```

```
DECL BOOL InitCtrlTool()
```

```
ENDDLL
```

```
' Initialize the CtrlTool Library
```

```
~^InitCtrlTool()
```

```
DLG 3D ON
```

```
STYLE% = DS_MODALFRAME | DS_SETFONT | WS_POPUP | WS_VISIBLE | WS_CAPTION
```

```
STYLE% |= WS_SYSMENU
```

```
DIALOG #2, 6, 15, 450, 170, "Balken-Demo", STYLE%, -12, "Ms Sans Serif"
```

```
CB DLG(2) 0 TO -1 CBX()
```

```
DEFPUSHBUTTON "Ende", 2, 330, 12, 100, 26, WS_TABSTOP | BS_DEFPUSHBUTTON
```

```
PUSHBUTTON "Test 1", 3, 330, 48, 100, 26, WS_TABSTOP
```

```
PUSHBUTTON "Test 2", 4, 330, 84, 100, 26, WS_TABSTOP
```

```
' Dialog control with progress class
```

```
CONTROL "", 100, "progress_class", WS_VISIBLE, 16, 35, 300, 60
```

```
ENDDIALOG
```

```
hProgress = DLGITEM (2, 100)
```

```
' or make a window with progress class
```

```
' hProgress = CreateWindowEx ($0, "progress_class", "", WS_CHILD | WS_VISIBLE, 16, 15, 300, 60, DLG(2),  
100, _INSTANCE, NULL)
```

```
SHOWDIALOG #2
```

```
' progress bar messages
```

```
PBM_DELTAPOS = WM_USER + 3
```

```
PBM_SETCOLOR = WM_USER
```

```
PBM_SETPOS = WM_USER + 2
```

```
PBM_SETRANGE = WM_USER + 1
```

```
PBM_SETSTEP = WM_USER + 4
```

```
PBM_STEPIT = WM_USER + 5
```

```
,
```

```
DO
```

```
SLEEP
```

```
LOOP UNTIL antw = IDCANCEL
```

```
,
```

```
,
```

```
FREEDLL 1
```

```
DLG 3D OFF
```

```
CLOSEDIALOG #2
```

```
END
```

```
RETURN
```

```

PROCEDURE CBX( hwnd, MSG, wParam, lParam%)
SELECT MSG
CASE WM_COMMAND
    SWITCH wParam
    ,
CASE IDCANCEL
    antw = IDCANCEL
    RETVAL FALSE
,
CASE 3 /* Test 1
    i = 0
    ' Set the progress bar color to red
    ~SendMessage (hProgress, PBM_SETCOLOR, 0, RGB (255, 0, 0))
    ' Reset the progress bar
    ~SendMessage (hProgress, PBM_SETPOS, 0, 0)
    ' Make sure the range is set
    ~SendMessage (hProgress, PBM_SETRANGE, 0, MAKELONG (20, 0))
    WHILE i < 20
        ' Add delay to simulate a process
        PAUSE 10
        ' Increments the position
        i = SendMessage (hProgress, PBM_DELTAPOS, i + 1, 0)
    WEND
    RETVAL FALSE
,
CASE 4 /* Test 2
    ' Reset the progress bar color, set the color to highlight
    ~SendMessage (hProgress, PBM_SETCOLOR, 1, 0)
    ' Reset the progress bar
    ~SendMessage (hProgress, PBM_SETPOS, 0, 0)
    ' Make sure the range is set
    uMin = 80
    uMax = 100
    ~SendMessage (hProgress, PBM_SETRANGE, 0, MAKELONG (uMax, uMin))
    ' Set the set value to 2
    ~SendMessage (hProgress, PBM_SETSTEP, 2, 0)
    i = 0
    WHILE i < uMax
        ' Add delay to simulate a process
        PAUSE 10
        ' Step the position
        i = SendMessage (hProgress, PBM_STEPIT, 0, 0)
    WEND
    RETVAL FALSE
ENDSWITCH
ENDSELECT
RETURN
,

```

BOOL OpenLongFileName (hwnd, flag%, title%, path%, ext%, filt%, file%)

WORD	hwnd&	/* Handle des Fensters	*/
LONG	flag%	/* Zeiger auf ein Flag	*/
LONG	title%	/* Zeiger auf Titel	*/
LONG	path%	/* Zeiger auf Pfad	*/
LONG	ext%	/* Zeiger auf Extension	*/
LONG	filt%	/* Zeiger auf Filter	*/
LONG	file%	/* Zeiger auf Filename	*/

Die OpenLongFileName()-Funktion ruft eine File-Auswahlbox nach Window 95-Art auf. Für den User besteht die Möglichkeit, einen Filenamen auszuwählen und zu laden.

Wird in der eigenen Applikation keine Dialogboxen verwendet, entfällt der Aufruf von DLG 3D ON/OFF.

Declaration:

```
DLL #1, "CTRLTOOL"  
DECL BOOL OpenLongFileName (w,l,l,l,l,l,l)  
ENDDLL
```

Parameter	Beschreibung	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
hwnd&	Handle des Fensters, in dem die Fileauswahlbox erscheinen soll.			
flag%	Zeiger auf ein 32 Bit-Wort, das folgende Werte verodert enthalten kann. OFN_FILEMUSTEXIST: Meldung bei nicht existierender Datei OFN_READONLY: Readonly-Schalter wird aktiviert. Ergebnis in Adresse Flag%.			
title%	Zeiger auf einen NULL-terminierten String, der den Titel der Dialogbox enthält. Wird NULL angegeben erscheint „Öffnen...“ als Titel.			
path%	Zeiger auf einen NULL-terminierten String, der den Pfadnamen enthält. Wird NULL angegeben, wird der aktuelle Pfad genommen.			
ext%	Zeiger auf einen NULL-terminierten String, der eine eventuelle Extension enthält. Beim Laden von Dateien nicht nötig. Hierbei immer NULL.			
filt%	Zeiger auf einen String, der die Filternamen enthält. Der String muß folgenden Aufbau haben: Filt\$ = „GFABasic (*.gfw)“ + CHR\$(0) + „*.gfw“ + CHR\$(0) Filt\$ = „Alle Dateien (*.*)“ + CHR\$(0) + „*.“ + CHR\$(0) + CHR\$(0) Der Filterstring muß unbedingt mit zwei CHR\$(0) enden. Mehrere Filterangaben mit Semikolon getrennt, wie „*.gfw;*.lst;*.res“, ist möglich. Wird NULL angegeben, so wird der standard Filter „*.“ angenommen.			
file%	Zeiger auf einen NULL-terminierten String, der einen Filenamen enthält. Es muß immer ein Zeiger auf einen Pufferstring übergeben werden, auch wenn er leer ist, also nur ein CHR\$(0) enthält			
Rückgabe:	1, wenn der OK-Button betätigt wird und eine Datei geladen wird. 0 bei Abbruch. Bei Auswahl eines Dateinamens, steht dieser an der Adresse file%. Ist das Flag OFN_READONLY gesetzt, so ist der Schalter „Mit Schreibschutz laden“ aktiv. Der Schalterzustand steht als Rückgabewert in der Adresse Flag%. Zustand = LONG{Flag%} & OFN_READONLY (1 oder 0).			

BOOL SaveLongFileName (hwnd, title%, path%, ext%, filt%, file%)

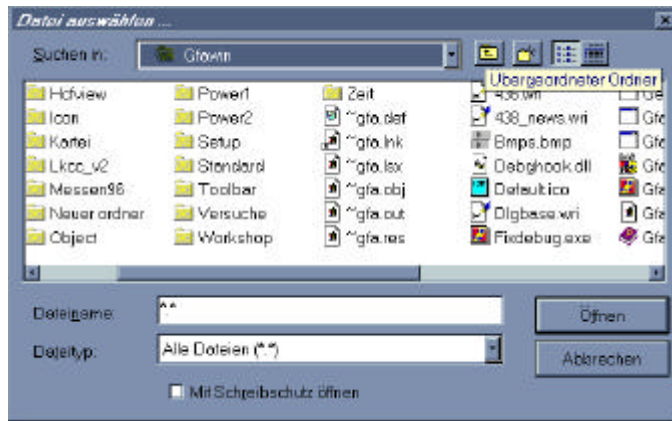
WORD	hwnd&	/* Handle des Fensters	*/
LONG	flag%	/* Zeiger auf ein Flag	*/
LONG	title%	/* Zeiger auf Titel	*/
LONG	path%	/* Zeiger auf Pfad	*/
LONG	ext%	/* Zeiger auf Extension	*/
LONG	filt%	/* Zeiger auf Filter	*/
LONG	file%	/* Zeiger auf Filename	*/

Die SaveLongFileName()-Funktion ruft eine File-Auswahlbox nach Window 95-Art auf. Für den User besteht die Möglichkeit, einen Filenamen auszuwählen und zu speichern.
Wird in der eigenen Applikation keine Dialogboxen verwendet, entfällt der Aufruf von DLG 3D ON/OFF.

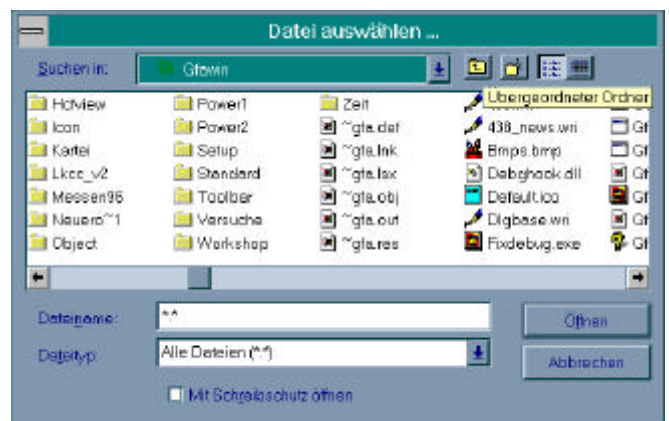
Declaration:

```
DLL #1, "CTRLTOOL"  
DECL BOOL SaveLongFileName (w,l,l,l,l,l)  
ENDDLL
```

Parameter	Beschreibung	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
hwnd&	Handle des Fensters, in dem die Fileauswahlbox erscheinen soll.			
flag%	Zeiger auf ein 32 Bit-Wort, das folgende veroderte Werte enthalten kann: OFN_OVERWRITEPROMPT: Meldung bei existierender Datei. OFN_READONLY: Wenn Flag gesetzt ist, wird beim Speichern darauf hingewiesen, daß die Datei schreibgeschützt ist. Beim Verlassen der Dialogbox wird der Schreibschutz wieder aufgehoben.			
title%	Zeiger auf einen NULL-terminierten String, der den Titel der Dialogbox enthält. Wird NULL angegeben erscheint „Speichern unter...“ als Titel.			
path%	Zeiger auf einen NULL-terminierten String, der den Pfadnamen enthält. Wird NULL angegeben, wird der aktuelle Pfad genommen.			
ext%	Zeiger auf einen NULL-terminierten String, der eine eventuelle Extension enthält. Wird eine Extension, z.B.: „BMP“, angegeben, so wird bei einem Dateinamen, in file% enthalten, an diesen angehängt und in das Editierfeld der Dialogbox geschrieben.			
filt%	Zeiger auf einen String, der die Filternamem enthält. Der String muß folgenden Aufbau haben: Filt\$ = „GFABasic (*.gfw)“ + CHR\$(0) + „*.gfw“ + CHR\$(0) Filt\$ = „Alle Dateien (*.*)“ + CHR\$(0) + „*.“ + CHR\$(0) + CHR\$(0) Der Filterstring muß unbedingt mit zwei CHR\$(0) enden. Mehrere Filterangaben mit Semikolon getrennt, wie „*.gfw;*.lst;*.res“, ist möglich. Wird NULL angegeben, so wird der standard Filter „*.“ angenommen.			
file%	Zeiger auf einen NULL-terminierten String, der einen Filenamen enthält. Es muß immer ein Zeiger auf einen Pufferstring übergeben werden, auch wenn er leer ist, also nur ein CHR\$(0) enthält			
Rückgabe:	1, wenn der OK-Button betätigt und eine Datei geladen wird. 0 bei Abbruch. Bei Auswahl eines Dateinamens, steht dieser an der Adresse file%.			



Window 95



Window 3.11

Beispiel: Dateiauswahlbox zum Laden von Dateien

Quelltext:

```
' Unter Window 3.x, NT lange Dateinamen nicht möglich
' Unter Window 95 lange Dateinamen wie gehabt.
' Aber immer Icons vor dem Ordner- oder Dateinamen.
'
```

```
DLL #1,"CTRLTOOL.DLL"
DECLL BOOL OpenLongFileName (w,l,l,l,l,l,l)
ENDDLL
```

```
' Erkennung der Version in der DLL
'
```

```
NULL$ = CHR$(0)
Title$ = "Datei auswählen ..." + NULL$
Path$ = CHR$(_DRIVE) + ":" + DIR$(0) + "\" + NULL$
```

```
Namebuffer$ = SPACE$(260)
' Es wird beim Laden kein Dateiname übergeben. Deshalb mit einem CHR$(0) füllen.
~Istrcpy (V:Namebuffer$, NULL$)
```

```
filt$ = "Alle Dateien (*.*)" + NULL$ + "*.*" + NULL$
filt$ = filt$ + "GFW-Datei (*.gfw)" + NULL$ + "*.gfw" + NULL$ + NULL$
```

```
flag% = OFN_FILEMUSTEXIST
```

```
DLG 3D ON
IF ^OpenLongFileName (NULL, V:flag%, V:Title$, V:Path$, NULL, V:filt$, V:Namebuffer$)
a$ = CHAR{V:Namebuffer$}
~MessageBox (NULL, V:Namebuffer$, "ausgewählte Datei:", MB_OK)
ENDIF
DLG 3D OFF
FREEDLL 1
'Ende des Beispiels
```

WORD PrinterSetup (hwnd)

WORD hwnd& /* Handle des Fensters */

Mit der PrinterSetup()-Funktion wird eine Dialogbox aufgerufen, die alle installierten Drucker in einer List-Box darstellt.

Man kann einen Drucker auswählen und zum Standard-Drucker machen.

Außerdem besteht die Möglichkeit mittels Optionen den Drucker einzustellen

oder mittels der Seitenformat-Buttons das Seitenformat direkt zu wählen.

Wird in der eigenen Applikation keine Dialogboxen verwendet, entfällt der Aufruf von DLG 3D ON/OFF.

Declaration:

```
DLL #1, "CTRLTOOL"  
DECL WORD PrinterSetup (w)  
ENDDLL
```

Parameter	Beschreibung	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
hwnd&	Handle des Fensters, in dem die Dialogbox erscheinen soll.			
Aufruf:	antw& = ^PrinterSetup (WIN (1)) Werden die Optionen des Druckers aufgerufen und mit OK bestätigt, wird der Button "Abrechen" in "Schließen" umbenannt.			
Rückgabe:	(antw&) bei Button "Abbrechen": bei Button "Schließen": bei Mißerfolg:	IDCANCEL (2) IDOK (1) 0		

Wird versucht aus zwei laufenden Programmen diese Dialogbox nacheinander aufzurufen, erscheint beim zweiten Aufruf eine Messagebox mit einer Mitteilung.

WORD GetPrinterOrientation ()

Mit der GetPrinterOrientation()-Funktion lässt sich vor einem Druckvorgang die Blattausrichtung des aktuellen Druckers erfragen.

Declaration:

```
DLL #1, "CTRLTOOL"  
DECL WORD GetPrinterOrientation ()  
ENDDLL
```

Parameter	Beschreibung	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
Rückgabe:	0 = Mißerfolg 1 = Hochformat 2 = Querformat			
Aufruf:	orient& = ^GetPrinterOrientation ()			

WORD SetPrinterOrientation (orient&)

WORD orient& /* Blattausrichtung */

Mit der SetPrinterOrientation()-Funktion lässt sich vor einem Druckvorgang die Blattausrichtung nur für diesen Ausdruck einstellen. Außerdem wird das Handle des DruckerDCs des aktuellen Druckers zurückgeliefert. Nach dem Druckvorgang wird die vorherige Blattausrichtung wieder hergestellt.
Diese Funktion ist eine Erweiterung der GFA-Basic-Funktion: hprint& = PrinterDC().

Declaration:

```
DLL #1, "CTRLTOOL"  
DECL WORD SetPrinterOrientation (w)  
ENDDLL
```

Parameter	Beschreibung	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
orient&	1 = Hochformat 2 = Querformat Wird orient& gleich Null gesetzt, bleibt die derzeitige Blattausrichtung bestehen. Es wird nur das Handle des Device-Context zurückgeliefert.			
Rückgabe:	Handle des Device-Context des aktuellen Standard-Druckers Null bei Mißerfolg.			
Aufruf:	hprint& = ^SetPrinterOrientation (1) /* Hochformat wird gesetzt. IF hprint& Druckvorgang ~DeleteDC (hprint&) ENDIF			

BOOL SystemInfo (hwnd, lp%)

WORD hwnd& /* Handle des Fensters */
LONG lp% /* Zeiger auf eine Struktur */

Mit der SystemInfo()-Funktion wird eine Dialogbox aufgerufen, die Benutzer- und Systeminformationen zeigt. Wird in der eigenen Applikation keine Dialogboxen verwendet, entfällt der Aufruf von DLG 3D ON/OFF.

Declaration:

```
DLL #1, "CTRLTOOL"  
DECL BOOL SystemInfo (w,l)  
ENDDLL
```

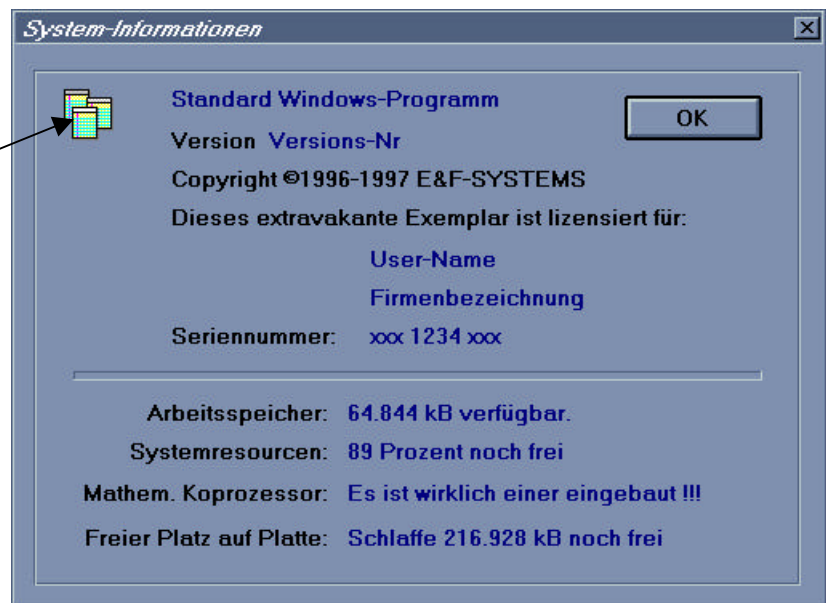
Parameter	Beschreibung	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
hwnd&	Handle des Fensters, in dem die Dialogbox erscheinen soll.			
lp%	Zeiger auf eine Programminfo-Struktur.			
	TYPE PROGRAMMINFO: - LONG lpstr1 Programm-Name - LONG lpstr2 Versions-Nummer - LONG lpstr3 User-Name - LONG lpstr4 Firmenbezeichnung - LONG lpstr5 Seriennummer - LONG lpstr6 Copyright-Meldung ENDTYPE Es können so bis zu 5 Strings an die SystemInfo-Dialogbox übergeben werden. Bei keiner Information prg.lpstrx = NULL setzen (x= 1-5).			

Rückgabe: Keine.

Beispiel: Aufruf einer SystemInfo-Dialogbox

```
DIM PROGRAMMINFO: prg.  
a$ = "Programm Name" + CHR$(0), prg.lpstr1 = V:a$  
a$ = "Versions-Nr" + CHR$(0), prg.lpstr2 = V:a$  
a$ = "User-Name" + CHR$(0), prg.lpstr3 = V:a$  
a$ = "Firmenbezeichnung" + CHR$(0), prg.lpstr4 = V:a$  
a$ = "xxx 1234 xxx" + CHR$(0), prg.lpstr5 = V:a$  
a$ = „1996, 1997 E&F-SYSTEMS, ptg.lpstr6 = V:a$  
~^SystemInfo (WIN (1), *prg.)
```

Programm-Icon wird an diese Stelle automatisch eingesetzt.



BOOL Ctl3dOn (hInstance)

WORD hInstance& /* Programm Instance */

Mit der Funktion Ctl3dOn() wird die 3D-Darstellung von Dialogboxen eingeschaltet. Es wird dazu die „CTL3D.DLL“ verwendet. GFA-Basic Funktion DLG 3D ON nicht nötig.

Declaration:

```
DLL #1, "CTRLTOOL"  
DECL BOOL Ctl3dOn (w)  
ENDDLL
```

Parameter	Beschreibung	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
hInstance&	_INSTANCE.			
Rückgabe:	BOOL: wenn erfolgreich, TRUE wird zurückgegeben, sonst FALSE			

BOOL Ctl3dOff (hInstance)

WORD hInstance& /* Programm Instance */

Mit der Funktion Ctl3dOff() wird die 3D-Darstellung von Dialogboxen ausgeschaltet. Es wird dazu die „CTL3D.DLL“ verwendet. GFA-Basic Funktion DLG 3D OFF nicht nötig.

Declaration:

```
DLL #1, "CTRLTOOL"  
DECL BOOL Ctl3dOff (w)  
ENDDLL
```

Parameter	Beschreibung	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
hInstance&	_INSTANCE.			
Rückgabe:	BOOL: wenn erfolgreich, TRUE wird zurückgegeben, sonst FALSE			

BOOL Ctl3dCtlOn (hwnd)

WORD hwnd& /* Fensterhandle */

Mit der Funktion Ctl3dCtlOn() wird ein mit der CreateWindow()-Funktion erstelltes Dialogelement in der eigenen Applikation in 3D dargestellt. Zum Beispiel Combobox- oder Editfenster in der Toolbar.

Declaration:

```
DLL #1, "CTRLTOOL"  
DECL BOOL Ctl3dCtlOn (w)  
ENDDLL
```

Parameter	Beschreibung	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
hwnd&	Handle des Dialogelements, das im eigenen Programm mit der 3D Darstellung versehen werden soll. Es muß vor der Initialisierung des MainWindows ^Ctl3dOn()-, und bei Programmende die ^Ctl3dOff()-Funktion aufgerufen werden.			
Rückgabe:	BOOL: wenn erfolgreich, TRUE wird zurückgegeben, sonst FALSE			

WORD CopyBmp (hBitmap, hpal)

WORD hwnd& /* Handle eines Bitmap */
WORD hpal& /* Handle einer Farbpalette */

Mit der CopyBmp()-Funktion wird ein Bitmap kopiert und das Handle des kopierten Bitmaps zurückgegeben.

Declaration:

```
DLL #1, "CTRLTOOL"  
DECL WORD CopyBmp (w,w)  
ENDDLL
```

Parameter	Beschreibung	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
hBitmap&	Handle des Fensters, das das Bitmap enthält.			
hpal&	Handle der Farbpalette, die zu diesem Bitmap gehört. Kann "NULL" sein, wenn keine vorhanden.			

Rückgabe: Handle auf das kopierte Bitmap.

Beispiel: Kopieren eines Bitmaps in die Zwischenablage:

```
hwnd& = WIN (1)  
IF OpenClipboard (hwnd&)  
    DEFMOUSE 2  
    ~EmptyClipboard()  
  
/* wenn eine Farbpalette vorhanden, kopiere sie zuerst in die Zwischenablage  
    ~SetClipboardData (CF_PALETTE, hmainpal&)  
  
/* erst dann das Bitmap  
    ~SetClipboardData (CF_BITMAP, ^CopyBmp(hBitmap&, hmainpal&))  
  
    ~CloseClipboard()  
    DEFMOUSE 0  
ENDIF
```

Dieses Beispiel kopiert ein vorhandenes Bitmap (Handle: hBitmap) in die Zwischenablage.

Um anderen Programmen den Zugang zu diesem Bitmap zu ermöglichen, muß das Handle der Bitmap auch nach Programmende im Windows vorhanden sein. Dies ist nur möglich, wenn das Bitmap kopiert und das kopierte Handle für die Zwischenablage zuständig ist.

Das originale Handle kann nun bei Beendigung des Programms mit FREEBMP hBitmap, oder mit ~DeleteObject (hBitmap) gelöscht werden.

Das kopierte Handle für das Bitmap in der Zwischenablage bleibt solange erhalten, bis der Inhalt der Zwischenablage gelöscht wird.

Wenn eine Farbpalette zu dem Bitmap vorhanden ist, muß diese auch in die Zwischenablage kopiert werden und beim Aufruf der Kopier-Funktion als zweiten Parameter mit angegeben werden.

LONG GetString (lp%, iString|)

LONG lp% /* Adresse eines Gesamt-Strings */
BYTE iString| /* Lage eines gesuchten Strings */

Die GetString()-Funktion gibt die Adresse eines gesuchten Strings zurück.
Diese Stringart kann verwendet werden, wenn man eine Reihe von konstanten Einträgen auswählen will und nur den Zeiger auf diesen ausgewählten String benötigt.

Declaration:

```
DLL #1, "CTRLTOOL"  
DECL LONG GetString (l,w)  
ENDDLL
```

Parameter	Beschreibung	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
lp%	Zeiger auf einen Gesamt-String, der aus einzelnen Teilstrings zusammengesetzt ist. Die Teilstrings müssen mit CHR\$(0) untereinander getrennt werden. Der Gesamtstring wird mit zwei CHR\$(0) abgeschlossen.			
iString	NULL basierende Lage des gesuchten Teilstrings.			
Rückgabe:	Zeiger auf den gesuchten Stringteil.			

Beispiel: Stringreihe zur Beschreibung von Toolbar-Buttons mittels Quick-Infos.

```
NULL$ = CHR$(0)  
gesString$ = "New" + NULL$ + "Open" + NULL$ + "Save" + NULL$  
gesString$ = gesString$ + "Print" + NULL$ + ..... + "Info" + NULL$ + NULL$  
  
lp% = V:gesString$  
  
teilString% = ^GetString (lp%, 1)  
  
puffer$ = SPACE$ (128)  
  
~lstrcpy (V:puffer$, teilString%)  
  
Print puffer$  
  
/* Ausgabe: "Open"
```

oder:

```
puffer$ = SPACE$ (128)  
  
~lstrcpy (V:puffer$, ^GetString (V:gesString$, 0))  
  
Print puffer$  
  
/* Ausgabe: "New"
```

LONG ChangeString (w|, lp%, AscFrom%, AscTo%)

Byte	w	/* Flag	
LONG	lp%	/* String-Adresse	*/
LONG	AscFrom%	/* String-Adresse	*/
LONG	AscTo%	/* String-Adresse	*/

Die ChangeString()-Funktion ändert oder fügt ein Ascii-Zeichen in einen gegebenen String.

Declaration:

```
DLL #1, "CTRLTOOL"  
DECL LONG ChangeString (w,l,l,l)  
ENDDLL
```

Parameter	Beschreibung	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
Flag	Das Flag kann folgende Werte haben: 1 = Ändern eines Ascii-Zeichens. 2 = Darstellung einer Zahl > 1000 mit Punkten.			
lp%	Adresse eines Null-Byte terminierten Strings.			
AscFrom%	Adresse eines Ascii-Zeichens, daß geändert werden soll. Bei Flag = 2 ist hier ein CHR\$(0) anzugeben.			
AscTo%	Adresse eines Ascii-Zeichens, das die Änderung, bei Flag = 1, oder Einfügung, bei Flag = 2, darstellt.			
Rückgabe:	Adresse des gänderten Strings.			

Beispiele:

1. Änderung eines Punktes in ein Komma

```
a$ = "10.12 cm" + CHR$(0)
```

```
a% = ^ChangeString (1, V:a$, ".", ",", ")
```

```
PRINT CHAR{a%}
```

Ausgabe: 10,12 cm

2. Umwandlung einer Zahl > 1000 in eine Darstellung mit Punkten

```
a$ = "1234756 Byte" + CHR$(0)
```

```
a% = ^ChangeString (1, V:a$, "", ".", ")
```

```
PRINT CHAR{a%}
```

Ausgabe: 1.234.756 Byte

LONG GetSystemTime (lp%)

LONG lp% /* Struktur-Adresse */

Die GetSystemTime()-Funktion liefert Angaben über das Datum und die Zeit zurück.

Declaration:

```
DLL #1, "CTRLTOOL"  
DECL LONG GetSystemTime (l)  
ENDDLL
```

Parameter	Beschreibung	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
lp%	Zeiger auf eine Struktur, die folgendermaßen deklariert werden muß. TYPE SYSTEMTIME: - WORD wYear - WORD wMonth - WORD wDayofWeek - WORD wHour - WORD wMinute - WORD wSecond - WORD wMilliseconds ENDTYPE wYear Jahreszahl zB.: 2000 wMonth Monat 1 – 12 wDayofWeek aktueller Wochentag, wobei 0 = Sonntag, 6 = Sonnabend. wHour Stunde wMinute Minute wSecond Sekunde wMilliseconds immer 0			

Rückgabe: Adresse der obigen Struktur mit Daten gefüllt.

LONG DayOfWeek (lp%)

LONG lp% /* Struktur-Adresse */

Die DayOfWeek()-Funktion liefert für das angegeben Datum den aktuellen Wochentag zurück.

Declaration:

```
DLL #1, "CTRLTOOL"  
DECL WORD DayOfWeek (l)  
ENDDLL
```

Parameter	Beschreibung	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
lp%	Zeiger auf eine SystemTime-Struktur, siehe unter GetSystemTime().			
Rückgabe:	0 für Sonntag, 1 für Montag,... 6 für Sonnabend.			

WORD MoonAge (tag&, monat&, jahr&)

```
WORD tag&          /* Word-Wert          */
WORD monat&        /* Word-Wert          */
WORD jahr&         /* Word-Wert          */
```

Die MoonAge()-Funktion liefert einen String zurück, der die Mondphase für das angegebene Datum aufzeigt.

Declaration:

```
DLL #1, "CTRLTOOL"
DECL WORD MoonAge (w,w,w)
ENDDLL
```

Parameter	Beschreibung	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
tag&	zu erfragender Tag.			
monat&	zu erfragender Monat.			
jahr&	zu erfragendes Jahr.			

Rückgabe: Wert zwischen 0 und 29, wobei folgendes gilt:

0, 29:	Neumond
1 – 6:	zunehmend, < Halbmond
7:	zunehmender Halbmond
8 – 13:	zunehmend, > Halbmond
14:	Vollmond
15 – 20:	abnehmend, > Halbmond
21:	abnehmender Halbmond
22 – 28:	abnehmend, < Halbmond

BOOL CopyFile (lp1%, lp2%)

```
LONG lp1%          /* Zeiger auf File 1  */
LONG lp2%          /* Zeiger auf File 2  */
```

Die CopyFile()-Funktion kopiert eine Datei.

Declaration:

```
DLL #1, "CTRLTOOL"
DECL BOOL CopyFile (l,l)
ENDDLL
```

Parameter	Beschreibung	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
lp1%	Zeiger auf einen String, der die zu kopierende Datei mit Laufwerks-, sowie Pfadangabe enthält.			
lp2%	Zeiger auf einen String, der die kopierte Datei mit Laufwerks-, sowie Pfadangabe enthält.			
Rückgabe:	Sind beide Dateinamen gleich, liefert die CopyFile()-Funktion NULL zurück, 1 bei Erfolg.			

BOOL DeleteFile (hwnd&, file%, flag!)

WORD	hwnd&	/* Fenster-Handle	*/
LONG	file%	/* Zeiger auf ein File	*/
WORD	flag!	/* Flag	*/

Die DeleteFile()-Funktion schiebt die zu löschende Datei unter Windows 95 in den Papierkorb. Unter Windows 3.11 und Windows NT wird die Datei gleich gelöscht.

Declaration:

```
DLL #1, "CTRLTOOL"  
DECL BOOL DeleteFile (w,l,w)  
ENDDLL
```

Parameter	Beschreibung	WINDOWS 3.11	WINDOWS 95	WINDOWS NT
hwnd&	Handle eines Fensters, in dem eine Löschmeldung erscheinen soll.			
file%	Zeiger auf einen Dateinamen, der gelöscht werden soll. Es ist unbedingt erforderlich, bei Aufruf dieser Funktion in das Laufwerk, bzw. Ordner zu wechseln, in dem die zu löschende Datei steht. Sonst erfolgt eine Fehlermeldung.			
flag!	TRUE: Dialogfeld zur Bestätigung wird eingeblendet. FALSE: Dialogfeld zur Bestätigung wird unterdrückt.			
Rückgabe:	1 bei Erfolg, 0 bei Mißerfolg.			

Nachtrag zu den Funktionen:

```
OpenLongFileName()  
SaveLongFileName()
```

Die Editierfunktion von Dateinamen direkt im Listefeld sind hinzugekommen. Durch zweimaliges Anklicken des Dateieintrages wird dieser in einem Editierfenster dargestellt. Siehe dazu die Funktionen zur originalen Windows 95 FileSelectBox (32-Bit-Version).

Verschieben einer selektierten Datei in den Papierkorb durch Betätigung der Delete-Taste läuft nur unter Windows 95.

Lange Dateinamen unter Windows NT sind nun auch möglich.