[e] forms and more

Technische Dokumentation
für Entwickler

Gültig ab Version 6.8.1

13.01.2014

Inhaltsverzeichnis

1. Einleitung

Die folgende Dokumentation ist für Programmierer gedacht, die die Formularverarbeitung ***[e] forms and more*** in eine bestehende Applikationsumgebung integrieren wollen.

Die Software besteht grundsätzlich aus zwei Komponenten - einem Editor zum Erstellen und Bearbeiten von Formularvorlagen - und einem Ausfüllprogramm zum Ausfüllen der vorbereiteten Vorlagen. Die Software steht einerseits als *ausführbare Applikation* zur Verfügung, andererseits als *Development-Kit* zur Einbindung in dritte Applikationen.

Alle wesentlichen Leistungen und Funktionen der Software sind in der zentralen dynamischen Bibliothek ***formdll.dll*** (16-Bit-Version) bzw. ***formdl32.dll*** (32-Bit-Version) konzentriert. Alle für die Einbindung notwendigen Funktionen sind in dieser Dokumentation beschrieben und können direkt aufgerufen werden.

In der Regel wird nur das Ausfüllen von Formularen in die Zielumgebung integriert. Der Formulareditor wird meist als eigenständige Applikation an den Kunden weitergegeben. Nichtsdestoweniger kann natürlich auch der Editor voll in die Rahmenapplikation eingebunden werden, wobei auch hier alle Bedienfunktionen von der Rahmenapplikation durch Aufruf gezielter DLL-Funktionen eingespeist werden können.

Die Rahmenapplikation stellt der Formularverwaltung - gleichgültig ob es sich um die Integration des Ausfüll-Moduls oder des Editors handelt - ein beliebiges Fenster zur Verfügung - das sogenannte Host-Fenster. Die Formularverwaltung öffnet in der *Client Area* des Host-Fensters ein rahmenloses, unsichtbares Verwaltungsfenster, in dem das gesamte Formulargeschehen abgewickelt wird. Dieses Fenster wird beim Initialisieren (*Form_Open_Expanded*) angelegt. Sein Handle wird an die Rahmenapplikation zurückgeliefert und bei allen weiteren Aufrufen, die diesem Formularfenster gelten, als Parameter mitgegeben.

Mittels dieser Dokumentation wird es einem erfahrenen Entwickler innerhalb kurzer Zeit gelingen, die Formularsoftware in eine Rahmenapplikation zu integrieren.

Um den Einstieg zu erleichtern, haben wir den C++-Quellcode der Rahmenapplikation zum Ausfüllen von Formularen ausführlich dokumentiert. Er liegt der Entwicklungsversion bei.

Das Waimea-Team wünscht viel Erfolg bei der Integration.

2. Start des Ausfüllers über Kommandozeile

Der Ausfüller von *[e] forms & more* bietet zur Verwendung als Anwendung im Stapelbetrieb (Batch) die Möglichkeit, über Kommandozeile gestartet zu werden. Für diesen Fall können das zu verwendende Formular sowie eine Reihe von Schaltern als Kommandozeilenparameter an den Ausfüller übergeben werden.

Der Programmstart via Kommandozeile beginnt wie üblich mit der Angabe des Programmnamens. Dieser ist normalerweise *filler.exe*. Der Programmname darf auch mit Pfad angegeben werden.

Nach dem Programmnamen können beliebig viele Kommandozeilenparameter folgen. Diese werden durch ein Leerzeichen voneinander getrennt. Der erste Parameter, der nicht mit einem Minuszeichen oder einem Schrägstrich / beginnen, wird als Dateiname einer Formularvorlage interpretiert und automatisch geöffnet.

Darüber hinaus können folgende Schalter gesetzt werden, die mit einem Minuszeichen oder einem / beginnen müssen:

/Minimize	Der Ausfüller wird minimiert gestartet. Das Programmsymbol erscheint in der Statuszeile, es wird aber kein Fenster eröffnet.
/Maximize	Der Ausfüller wird maximiert gestartet.
/Resize	Der Ausfüller wird automatisch in normaler Fenstergröße gestartet.
/HideMenu	Der Ausfüller wird ohne Menüleisten gestartet.
/HideButtons	Der Ausfüller wird ohne Buttonzeile gestartet.
/HideStatus	Der Ausfüller wird ohne Statuszeile gestartet.
/NoCloseOnEscape	Die Taste ESC schließt nicht wie sonst üblich das aktuelle Formularfenster.
/WindowHeight	Höhe des Anwendungsfensters in Pixeln
/WindowWidth	Breite des Anwendungsfensters in Pixeln

/DaccWatchLimit	Alle ODBC-Datenzugriffe werden mit dem Flag ACCFLAG_WATCH_SELECT_LIMIT ausgeführt. Dies führt dazu, dass ohne Rückfrage nicht mehr als eine limitierte Anzahl Sätze selektiert wird. Damit kann auf langsamen Übertragungsstrecken das unnötige Übertragen großer Datenmengen verhindert werden.
/ButtStat=file	Der Ausfüller liest die Buttons und die Statuszeile nicht aus der vordefinierten Datei sondern aus der hier angegebenen. Die Datei ist strukturell eine Profile-Datei.
/Title=xxx	Der Titel des Hauptfensters wird mit der angegebenen Zeichenfolge überschrieben.
/Exit	Der Ausfüller wird unverzüglich beendet.

/Perform=file Es wird das Makro ausgeführt, welches sich in der angegebenen Makrodatei *File* befindet. Das Makro bezieht sich auf das letzte geöffnete (also auf das aktive) Formular. Das Makro muss in der der Skriptsprache *Wincula* verfaßt sein.

Alle angegebenen Schalter werden von links nach rechts abgearbeitet. Wenn ein Schalter oder ein Dateiname Leerzeichen enthält, so muss dieser in Anführungszeichen "" eingeschlossen werden. Der Delimiter, also das Minuszeichen bzw. der Schrägstrich, muss im Falle eines Schalters mit eingeschlossen werden.

Beispiel

Die folgende Kommandozeile startet den Ausfüller minimiert, öffnet das Formular *aaa.fvl*, wendet die Makrodatei *bbb.fmc* an und beendet den Ausfüller sogleich wieder:

filler aaa.fvl /minimize /perform=bbb.fmc /exit

Der Start des Ausfüllers auf Kommandozeilenebene ist insbesondere für die Einbindung in Systemumgebungen gedacht, in denen der direkte Zugriff auf DLL-Schnittstellen nicht möglich ist.

Für weitergehende Kontrolle über den Ausfüller sollte auf eine Einbindung über Interface-Routinen zurückgegriffen werden. Diese wir in den folgenden Kapiteln erläutert.

3. Das Interface

Das Programmier-Interface der Formularsoftware *[e]forms and more* besteht aus einer Vielzahl von Funktionen, die in einer DLL zur Verfügung gestellt werden.

Die Interface-Funktionen können sich auf verschiedene Komponenten des Formulars beziehen. Insgesamt stehen folgende Funktionsgruppen zur Verfügung:

- Funktionen für das gesamtes Formular
- Funktionen für einzelne Seiten des Formulars
- Funktionen für einzelne Objekte des Formulars.

Jede der Funktionen wird mit einem Handle versorgt, der die entsprechende Komponente eindeutig identifiziert.

Das gesamte Formular, wird über den Formular-Handle angesprochen, der beim Öffnen des Formulars zurückgeliefert wird (siehe *Form_Open_Expanded)*.

Eine einzelne Seite wird über einen *Page-Handle* angesprochen. Dieser kann mit der Funktion *Form_GetPageHandle* für jede Seite auf Basis der Seitennummer ermittelt werden.

Einzelne Objekte einer Seite werden über einen *Objekt-Handle* angesprochen. Dieser kann für jedes Objekt mittels der Funktion *Form_FindNextObject* auf Basis des Objektnamens und –Typs ermittelt werden.

3.1. Aufruf- und Namenskonventionen

Damit die Formularsoftware *[e] forms and more* in möglichst viele Umgebungen problemlos eingebunden werden kann, werden alle Schnittstellen von *[e] forms and more* für unterschiedliche Aufrufkonventionen bereitgestellt. Je nach dem, welche Aufrufkonventionen das Entwicklungssystem, in das die Formularsoftware eingebunden werden soll, unterstützt, kann auf die jeweilige Aufrufkonvention zurückgegriffen werden.

Zur Zeit werden folgende Aufrufkonventionen unterstützt:

- stdcall
- cdecl

Funktionen, die mit der Aufrufkonventionen ***stdcall*** compiliert sind, entfernen selbst ihre Aufrufparameter vom Stack, während dies bei Funktionen, die mit ***cdecl*** compiliert sind, der Pro-

grammcode des aufrufenden Programmzweigs erledigt. Dadurch sind beide Konventionen zueinander inkompatibel.

Leider befinden sich auf dem Markt sowohl Entwicklungssysteme, die die Aufrufkonvention ***cdecl*** verwenden (z.B. *Smalltalk)*, als auch solche, die ***stdcall*** benötigen (z.B. *Visual Basic)*. Um möglichst viele Programmierumgebungen unterstützen zu können, werden alle Schnittstellen der Formularsoftware in beiden Konventionen bereitgestellt.

Um die Routinen für beide Konventionen unterscheiden zu können, wurde eine Namenskonvention eingeführt. Die Namen der Routinen für die Konvention *stdcall* beginnen alle mit einem großen *"S".*

Ferner beginnen die Namen der Funktionen, die sich auf des gesamte Formular beziehen, mit dem Prefix *Form...,* die Funktionen,m die sich auf einzelne Seiten beziehen mit dem Prefix *FormPage...* und die Funktionen, die sich auf einzelne Objekte beziehen beginnen mit dem Prefix *FormObj...*

Insgesamt gibt es also folgende Namensprefixe:

	Aufrufkonvention **cdecl**	Aufrufkonvention **stdcall**
Gesamtes Formular	Form_	SForm_
Einzelne Seite	FormPage_	SFormPage_
Einzelnes Objekt	FormObj_	SFormObj_

Der Übersichtlichkeit halber werden in dieser Dokumentation nur die ***cdecl-***Schnittstellen beschrieben. Die Beschreibung kann jedoch eins zu eins auch für die ***stdcall-***Schnittstellen verwendet werden. Es muss lediglich dem Namen der Funktion ein großes ***S*** vorangestellt werden.

Im folgenden werden noch die verschiedenen Entwicklungsumgebungen genannt und welche Aufrufkonventionen entsprechend verwendet werden sollten.

Visual Basic

Wenn *[e] forms and more* in eine *Visual Basic* Umgebung eingebunden wird, muss in jedem Fall die Konvention *stdcall* verwendet werden. Es müssen also diejenigen Routinen verwendet werden, deren Namen mit einem großen ***S*** beginnt.

Der Einbindung in ein *Visual Basic Projekt* ist ein eigenes Kapitel in dieser Beschreibung gewidmet.

C oder C++

Für die Einbindung von *[e] forms and more* in ein C++ oder C-Projekt können beide möglichen Aufrufkonventionen verwendet werden. Der Compiler unterscheidet die jeweilige Konvention durch die *Prototypes*, welche in der Header-Datei definiert sind.

Auch der Einbindung in ein C++ oder C-Projekt ist ein eigenes Kapitel in dieser Beschreibung gewidmet.

Andere Umgebungen

Wenn *[e] forms and more* in eine andere Umgebung eingebettet wird, entnehmen Sie bitte der Dokumentation der jeweiligen Entwicklungsumgebung, welche Aufrufkonvention erwartet wird.

3.2. Funktionsbeschreibungen

Die folgenden Kapitel beschreiben alle Funktionen, die vom Interface bereitgestellt werden. Die einzelnen Funktionen sind in alphabetischer Reihenfolge angeordnet. Die Beschreibung jeder Funktion besteht aus einem Kopf, welcher den Namen und die Parameter der Funktion sowie die Typen der Parameter aufzählt und einem Rumpf, welcher die Funktion beschreibt.

Wenn die Funktion einen Funktionswert zurückliefert, wird dieser nach dem Rumpf beschrieben. Danach folgen Verweise auf verwandte Funktionen, falls solche existieren.

Bitte lesen Sie unbedingt auch das Kapitel *Veraltete Funktionen.* Hier werden alle Funktionen beschrieben, die mittlerweile durch neuere ersetzt worden sind.

4. Formularfunktionen

Form_ActivateFormWindow

```
HWND      hwndForm
```

Diese Funktion aktiviert ein Formularfenster. Die Wirkung dieser Funktion hängt davon ab, ob sich das Formular im Ausfüllmodus oder im Ediermodus befindet:

- Im Ediermodus erhält das Formularfenster den Fokus.
- Im Ausfüllmodus erhält das augenblicklich aktive Eingabeobjekt den Fokus.

Die Funktion wird vornehmlich aufgerufen, wenn im Rahmen einer MDI-Applikation ein MDI-Fenster, in dem ein Formular abgewickelt wird, aktiviert wird.

Die Funktion liefert keinen Funktionswert zurück.

Form_AppendModule

```
HWND      hwndForm
char *    FileName
```

Die Funktion fügt einen Text aus der Bausteinverwaltung in das augenblicklich aktive Eingabeobjekt des Formulars ein. Sie kann sowohl im Ediermodus als auch im Ausfüllmodus aufgerufen werden.

hwndForm bezeichnet das Formularfenster, auf welches sich der Aufruf der Bausteinverwaltung bezieht.

FileName bezeichnet den Namen des einzufügenden Bausteins. Wenn der Wert NULL angegeben wird, wird der Bausteinname mittels eines Dialoges der Bausteinverwaltung angefragt. Der Dialog ermöglicht nicht nur die Auswahl eines Bausteins, sondern auch die Pflege der Bausteinbibliothek.

Die Funktion liefert keinen Funktionswert zurück.

Form_AppendObjSet

HWND hwndForm
char * FileName
longMode

Die Funktion fügt alle Objekte der ersten Seite eines externen Formulars in die aktuelle Seite des geöffneten, durch *hwndForm* spezifizierten, Formulars ein. Der Name des Vorlagen-Formulars wird auf dem Parameter *FileName* übergeben. Mode ist für zukünftige Zwecke gedacht und muss mit 0 vorbesetzt werden.

Der Funktionswert ist vom Typ *integer*.

IDOK Die Objekte wurden fehlerfrei übertragen.
IDCANCEL Es ist ein Fehler aufgetreten.

Verwandte Funktionen
Form_SplitObjSet

Form_ArrangeSelectedObjects

HWND hwndForm
int Modus
long Distanz

Die Funktion ordnet alle markierten Objekte eines Formulars an, richtet diese aus oder gleicht die Größe von Objekten einander an. Diese Funktion ist im Ausfüllmodus ohne Wirkung. Bei Aufruf der Funktion werden alle sekundär markierten Objekte am primär markierten Objekt orientiert.

Der Parameter Modus bestimmt, welche Operationen ausgeführt werden. Es können ein oder mehrere der folgenden Werte angegeben werden:

ARR_ALIGNLEFT	untereinander am linken Rand ausrichten
ARR_ALIGNRIGHT	untereinander am rechten Rand ausrichten
ARR_ALIGNTOP	nebeneinander am oberen Rand ausrichten
ARR_ALIGNBOTTOM	nebeneinander am unteren Rand ausrichten
ARR_ALIGNVERTCENTER	untereinander mittig zentriert ausrichten
ARR_ADJUSTHEIGHT	die Höhe der Objekte beim Ausrichten angleichen
ARR_ADJUSTWIDTH	die Breite der Objekte beim Ausrichten angleichen

ARR_ADJUSTVERTDIST	Objekte senkrecht im Abstand Distanz anordnen
ARR_ADJUSTHORIDIST	Objekte waagerecht im Abstand Distanz anordnen

Mehrere Angaben sind durch ein logisches oder voneinander zu trennen. Naturgemäß widersprechen sich manche der Attribute. Attribute, die sich widersprechen, dürfen nicht gleichzeitig angegeben werden.

Die Funktion liefert keinen Funktionswert zurück.

Form_ChangeObjectsType

HWND	hwndForm
int	newType
long	lParam
int	flags

Diese Funktion wandelt Objekte der aktuellen Seite in Objekte eines anderen Typs um.

Nicht jede Umwandlung ist möglich. Es können nur sinnvolle Umwandlungen durchgeführt werden. Es macht z.B. keinen Sinn, Eingabefelderin Linien zu verwanden.

newType Gibt den neuen Objekttyp an.

Alle möglichen Objekttypen sind im Anhang OBJEKTTYPEN zusammengefasst.

0 bedeutet, dass der neue Objekttyp mittels eines Dialog angefragt wird.

lParam Ist für spätere Zwecke reserviert und muss immer 0 sein.

flags ist eine Liste von Bits folgender Optionen:

FM_SELECTED	Operation nur für markierte Objekte ausführen
FM_INPUT	Operation nur für Eingabeobjekte ausführen
FM_GROUPED	Operation nur für gruppierte Objekte ausführen
FM_BEEP	Im Fehlerfall einen Klang ausgeben
FM_UNDO	Ein UnDo-Item absetzen
FM_EDITMODE	Funktion ablehnen, falls nicht der Bearbeitungsmodus eingeschaltet ist
FM_REFRESH	Änderungen sofort anzeigen
FM_NOTIFY	Benachrichtigung über die Aktion an das Hostfenster senden

Die Funktion liefert keinen Funktionswert zurück.

Form_CheckPasswords

HWND hwndForm
int Modus

Diese Funktion dient dazu, eine Passwortprüfung für ein Formular durchzuführen. Der Parameter Modus definiert, welcher Passworttyp überprüft werden soll:

PW_EDIT Es wird das Passwort zur Berechtigung des Veränderns einer Formularvorlage überprüft (erfolgt z.B. beim Öffnen einer Vorlage im Formulareditor).

Wenn für das Formular das entsprechende Passwort nicht definiert ist, kehrt die Funktion ohne weitere Anfrage mit dem Wert 1 (Passwort richtig) zurück.

Wenn das Formular mit einem Passwort geschützt ist, wird ein Dialog eröffnet, in dem der Benutzer das Passwort eingeben kann. Gibt der Benutzer das richtige Passwort ein, kehrt die Funktion ebenfalls mit dem Wert 1 (Passwort richtig) zurück.

Andernfalls wird der Benutzer auf das falsche Passwort hingewiesen und erneut nach dem Passwort gefragt, bis ein richtiges Passwort eingegeben wird.

Wenn der Benutzer den Dialog abbricht, kehrt die Funktion mit dem Wert 0 (Passwort wurde nicht korrekt eingegeben) zurück.

Funktionswert vom Typ *integer*

1 Es wurde ein korrektes Passwort eingegeben
oder das Formular ist nicht passwortgeschützt.

0 Es wurde kein korrektes Passwort eingegeben.
Der Dialog wurde vom Benutzer abgebrochen.

Form_CheckTranslateAccel

MSG * Message

Diese Funktion kann in der Message-Loop der Rahmen-Applikation aufgerufen werden. Sie prüft, ob die angegebene Windows-Message in der Anwendung mittels der Funktion *TranslateAccelerator* modifiziert werden sollte oder nicht.

Insbesondere die Messages WM_KEYUP, WM_KEYDOWN and WM_CHAR für Edit-Control-Fenster sollten nicht modifiziert werden, damit diejenigen Tasten, die innerhalb von Edit-Controls eine spezifische Bedeutung haben, einwandfrei funktionieren können.

Funktionswert vom Typ *integer*

1 Die angegebene Message kann ohne weiteres mittels der Funktion *TranslateAccelerator* modifiziert werden.

0 Die angegebene Message sollte nicht mittels der Funktion *TranslateAccelerator* modifiziert werden.

Form_Close

HWND hwndForm

Diese Funktion schließt ein geöffnetes Formularfenster. Alle von diesem Formular belegten Systemresource werden wieder freigegeben. Das Parent-Window kann für ein anderes Formular weiterverwendet werden. Wird das Parent-Window geschlossen, so ist es nicht nötig *Form_Close* aufzurufen, da impliziet ein *Form_Close* durchgeführt wird.

Die Funktion liefert keinen Funktionswert zurück.

Verwandte Funktionen
Form_Open_Expanded

Form_CreateAutoForm

HWND	hwndForm
char *	FieldDescList
AUTOFORMDESIGN *	Design

Diese Funktion erzeugt ein einseitiges Formular. *hwndForm* muss zuvor mittels der Routine *Form_Open_Expanded* angelegt oder eröffnet worden sein.

Das Aussehen sowie alle Objekte des Formulars werden automatisch nach den Angaben in den Parametern *FieldDescList* und *Design* generiert. Die Struktur Design muss vorher manuell oder mit Hilfe der Funktion *Form_PresetAutoFormDesign* vorbesetzt werden.

Die zu erzeugenden Felder werden auf dem Zeiger *FieldDescList* übergeben. *FieldDescList* setzt sich aus einer oder mehreren Feldbeschreibungen zusammen. Jede Feldbeschreibung muss mit dem Namen des anzulegenden Feldes beginnen. Einzelne Felder werden durch TAB (\t) getrennt, Eigenschaften eines Feldes mit einem Vertikaltabulator (\v).

Folgende Eigenschaften können gelistet werden:

Prompt	Gibt den Beschreibungstext vor dem Feld an.
DataType	Gibt den Datentyp des Feldes an: I Ganzzahl (Integer) F Gleitkommazahl (Float) T Uhrzeit (Time) D Datum (Date) Wird kein Datentyp spezifiziert, so ist das Feld vom Typ ‚Text'.
StateBits	Gibt einen besonderen Status für das Feld an: R Das Feld kann zwar gelesen jedoch nicht geändert werden.
Format	Gibt die Formatieranweisung für das Feld an. Nähere Informationen zur Formatbeschreibung enthält eine gesonderte Dokumentation.
MaxLng	Gibt die maximale Länge in Zeichen einer Eingabe in dem Feld an. Wird keine Länge festgelegt, so ist der Maximalwert 9999.

Der Funktionswert ist vom Typ *integer*.

IDOK Das Formular wurde fehlerfrei generiert.
IDCANCEL Es ist ein Fehler aufgetreten, das Formular konnte nicht generiert werden.

Verwandte Funktionen

Form_CreateDatabaseForm
Form_PresetAutoFormDesign

Form_CreateDatabaseForm

HWND	hwndForm
char *	DataSource
char *	UserId
char *	Password
char *	Table
AUTOFORMDESIGN *	Design
long	AccessFlags
long	Flags

Diese Funktion erzeugt ein einseitiges Formular mit integriertem Datenbankzugriff. . *hwndForm* muss zuvor mittels der Routine *Form_Open_Expanded* angelegt oder eröffnet worden sein.

Das Aussehen sowie alle Objekte des Formulars werden automatisch aufgrund der angegebenen Datenzugriffsbeschreibung (*DataSource, UserId, Password*) und den Angaben in dem Parameter *Design* generiert.

DataSource gibt den Namen der Datenquelle an. *UserId* und *Password* müssen einen Zugriff auf die Datenquelle ermöglichen. Table gibt den Namen der Tabelle oder Abfrage an, auf welche zugegriffen werden soll. Die Struktur *Design* muss vorher manuell oder mit Hilfe der Funktion *Form_PresetAutoFormDesign* vorbesetzt werden.

Mit dem Parameter *AccessFlags* werden die Datenzugriffsmöglichkeiten bestimmt (siehe *FormObj_SetLongProperty* unter FORMOBJ_PROP_ACCESS_FLAGS). Flags ist für zukünftige Zwecke gedacht und muss mit Null vorbesetzt werden.

Der Funktionswert ist vom Typ *integer*.

IDOK	Das Formular wurde fehlerfrei generiert.
IDCANCEL	Es ist ein Fehler aufgetreten, das Formular konnte nicht generiert werden.

Verwandte Funktionen

Form_CreateAutoForm
Form_PresetAutoFormDesign
FormObj_SetLongProperty

Form_CreateObject

```
HWND   hwndForm
int    objectType
long   coFlags
int    flags
```

Diese Funktion legt ein neues Objekt auf der aktuellen Seite eines Formulars an.
Wenn eine Zone selektiert ist, wird das Objekt in der Zone angelegt.

objectType Gibt den Typ des neuen Objekts an.

Alle möglichen Objekttypen sind im Anhang OBJEKTTYPEN zusammengefasst.

0 bedeutet, dass der neue Objekttyp mittels eines Dialogs angefragt wird.

coFlags Bitliste – es lassen sich folgende Optionen wählen:

FORM_CO_OPEN_DIALOG	Für das neu angelegte Objekt wird der Dialog zum Einstellen der Eigenschaften automatisch geöffnet.
FORM_CO_UNSELECT	Alle selektierten Objekte werden auf nicht selektiert gesetzt, nachdem das neue Objekt angelegt ist. Das neue Objekt wird primär selektiert.
FORM_CO_SETBYMREF	Das neu angelegte Objekt wird mit den Eigenschaften des Master-Referenz-Objekts vorbesetzt.
FORM_CO_NO_MODECHECK	Es findet keine Prüfung statt, ob der Arbeitsmodus FORMMODE_EDIT eingeschaltet ist. Dies ermöglicht ein Anlegen neuer Objekte auch im Ausfüllmodus.
FORM_CO_NO_SELECT	Entgegen der sonst üblichen Verfahrensweise wird das neu angelegte Objekt nicht selektiert.

flags Bitliste – Es lassen sich folgende Optionen wählen:

FM_BEEP	Im Fehlerfall einen Klang ausgeben
FM_UNDO	Ein UnDo-Item absetzen
FM_EDITMODE	Funktion ablehnen wenn das Formular nicht im Edit-Modus ist
FM_REFRESH	Sofortige Anzeige des neuen Objekts
FM_NOTIFY	Notification senden

Funktionswert vom Typ *HFOB (Formularobjekt-Handle)*

hFob Das Objekt wurde erfolgreich angelegt.
Der Handle des Objekts wird als Funktionswert zurückgeliefert.

0 Es ist ein Fehler aufgetreten oder ein Dialog wurde abgebrochen.

Diese Funktion ersetzt folgende ältere Funktionen
Form_CreateNewObject
Form_CreateNewObjectExtended

Die älteren Funktionen sollten nicht mehr verwendet werden.

Form_DataAccess

HWND hwndForm
int Mode
long lParam
long Flags

Diese Funktion ermöglicht den Datenzugriff auf externe Datenbanken. Ist das aktuell selektierte Objekt mit einem Datenzugriffsobjekt verbunden, so wird der Zugriff an dieses Objekt weitergeleitet. Der Zugriff bezieht sich also auf die im assoziierten Datenzugriffsobjekt eingestellte Tabelle oder Abfrage. Ist kein Objekt selektiert, oder kein Datenzugriffsobjekt mit dem selektierten Objekt verbunden, so wird der gewünschte Zugriff von dem mit dem Formular verbundenen Datenzugriff ausgeführt.

Mit dem Parameter *Mode* wird bestimmt, welche Art des Datenzugriffs ausgeführt werden soll. Der Wert von *lParam* hat abhängig von Mode eine unterschiedliche Bedeutung. Mit *Flags* kann das Verhalten der Funktion gesteuert werden.

Folgende Angaben für den Parameter *Mode* sind möglich:

FACC_SELECT_EXACT	Es werden alle Sätze selektiert, die mit den aktuell gemachten Eingaben (ein oder mehrere Felder) exakt übereinstimmen.
FACC_SELECT_LEAD	Es werden alle Sätze selektiert, die in den entsprechenden Feldern mit den aktuellen Eingaben beginnen. Beispiel: ‚Schmidt' selektiert ‚Schmidt', ‚Schmidtmann' und ‚Schmidtchen'
FACC_SELECT_STD_MASK	Es wird ein Dialog eröffnet, in dem umfassende Kriterien zur Selektion von Datensätzen eingegeben werden können.
FACC_SELECT_ALL	Es werden alle Datensätze selektiert, d.h. es wird die aktuelle Selektion (Filter) aufgehoben.
FACC_SELECT	Es wird aufgrund zuvor gesetzter Kriterien erneut selektiert (ohne die Bedingung(en) zu ändern). Gelöschte und geänderte Sätze fallen dadurch heraus, neu erfasste Sätze kommen möglicherweise hinzu.
FACC_UNSELECT	Die aktuelle Selektion wird aufgehoben. Danach befindet sich kein Datensatz mehr im Zugriff.
FACC_GOTO_FIRST	Sprung auf das erste Feld des aktuellen Datensatzes.
FACC_GOTO_LAST	Sprung auf das letzte Feld des aktuellen Datensatzes.

FACC_ORDER_ASC	Die aktuelle Selektion wird aufsteigend sortiert.
FACC_ORDER_DESC	Die aktuelle Selektion wird absteigend sortiert.
FACC_NAVIGATE_NEXT	Es wird auf den nächsten Datensatz innerhalb der Selektion geblättert.
FACC_NAVIGATE_PREV	Es wird auf den vorigen Datensatz innerhalb der Selektion geblättert.
FACC_NAVIGATE_FIRST	Es wird auf den ersten Datensatz der Selektion geblättert.
FACC_NAVIGATE_LAST	Es wird auf den letzten Datensatz der Selektion geblättert.
FACC_RECORD_UPDATE	Es wird der aktuelle Datensatz inklusive der durchgeführten Änderungen in die Datenbank zurückgeschrieben.
FACC_RECORD_INSERT_EMPTY	Es wird ein leerer Datensatz in die Datenbank eingefügt. Dieser wird zum aktuellen Datensatz.
FACC_RECORD_INSERT_CLONE	Es wird eine Kopie des aktuellen Datensatzes in die Datenbank eingefügt. Dieser wird zum aktuellen Datensatz.
FACC_RECORD_INDERT_CURRENT	Es wird ein neuer Datensatz in der Datenbank angelegt. Die aktuellen Eingaben werden für diesen Datensatz übernommen. Dieser wird der aktuelle Datensatz.
FACC_RECORD_DELETE	Der aktuelle Datensatz wird in der Datenbank gelöscht.
FACC_RECORD_RELOAD	Der aktuelle Datensatz wird erneut aus der Datenbank gelesen. Nicht gespeicherte Änderungen gehen dabei verloren.

FACC_MATCHMODE_ENTER	Es wird in den Modus zur Eingabe von Suchbegriffen umgeschaltet.
FACC_MATCHMODE_CANCEL	Der Modus zur Eingabe von Suchbegriffen wird abgebrochen, die aktuell gültige Selektion wird nicht verändert.
FACC_MATCHMODE_CONFIRM	Die Eingabe von Suchbegriffen wird abgeschlossen und bestätigt. Es werden alle Sätze selektiert, die in den entsprechenden Feldern mit den aktuellen Eingaben beginnen.
FACC_MATCHMODE_TOGGLE	Der Editiermodus und der Modus zur Eingabe von Suchbegriffen wird gewechselt. Im Editiermodus ist dies identisch mit FACC_MATCHMODE_ENTER im Suchbegriffeingabemodus mit dem Modus FACC_MATCHMODE_CONFIRM.
FACC_SET_CHANGEFLAG	Dieser Aufruf setzt die Eigenschaft *Geändert* des aktuell eingestellten Datensatzes. lParam gibt den Wert an (Input): 1 Datensatz wurde geändert. 0 Datensatz wurde nicht geändert.
FACC_GET_CHANGEFLAG	Dieser Aufruf fragt die Eigenschaft *Geändert* des aktuell eingestellten Datensatzes ab. Returnwert: 1 Datensatz wurde geändert. 0 Datensatz wurde nicht geändert.
FACC_GET_TOTAL_RECORDS	Dieser Aufruf fragt die Anzahl der Datensätze innerhalb der aktuellen Selektion ab. Returnwert:

	Anzahl der selektierten Datensätze
FACC_GET_CURRENT_RECORD	Dieser Aufruf fragt die Nummer des eingestellten Datensatzes innerhalb der aktuellen Selektion ab. Returnwert: > 0 Nummer des aktuellen Datensatzes (ab 1 gezählt) == 0 Es ist kein Datensatz eingestellt.
FACC_GET_MATCHMODE	Dieser Aufruf fragt den Status des Modi zur Eingabe von Suchbegriffen ab. Returnwert: 1 Suchbegriffeingabe ist eingeschaltet. 0 Suchbegriffeingabe ist ausgeschaltet.
FACC_GET_WHERE_CLAUSE_LNG	Dieser Aufruf fragt die Anzahl Zeichen des aktuellen Suchkriteriums ab. Diese Option dient dazu, bei einer nachfolgenden Abfrage des Suchkriteriums selbst, genügend Speicher bereitzustellen. Returnwert: > 0 Anzahl Zeichen des aktuellen Suchkriteriums. == 0 Es ist kein Suchkriterium eingestellt.
FACC_GET_ORDERBY_CLAUSE_LNG	Dieser Aufruf fragt die Anzahl Zeichen des aktuellen Sortierkriteriums ab. Diese Option dient dazu, bei einer nachfolgenden Abfrage des Sortierkriteriums selbst, genügend Speicher bereitzustellen. Returnwert: > 0 Anzahl Zeichen des aktuellen Sortierkriteriums. == 0 Es ist kein Sortierkriterium eingestellt.

FACC_PROFILE_WRITE	Es werden alle aktuellen und interessanten Informationen (z.B. Such- und Sortierkriterium, sowie das gesamte Selektionsstatement) des Datenzugriffsobjektes in eine Datei geschrieben. Die Datei ist im Aufbau gleich einer Profile-Datei. Der Name der Datei wird auf lParam übergeben. Die Sektion heißt [ODBCACC]. lParam (Input): Zeiger auf eine Zeichenkette mit dem Namen der zu schreibenden Datei.

Folgende Angaben für den Parameter Flags sind möglich:

ACCFLAG_NO_CONFIRM	Diese Option unterdrückt jegliche Sicherheitsabfragen beim Datenzugriff (z.B. Ändern oder Löschen von Datensätzen) (Stummschaltung).
ACCFLAG_NO_TEMPDISP	Diese Option unterdrückt die Anzeige des kurzzeitigen Bestätigungsfensters (“Datensatz wurde gespeichert”, ...).
ACCFLAG_NO_BEEP	Diese Option unterdrückt das akustische Signal, welches bei Nichtdurchführbarkeit einer Funktion üblicherweise ertönt (z.B. Blättern vor den ersten Datensatz, ...).
ACCFLAG_NO_MACROCALL	Diese Option unterdrückt die Ausführung irgendwelcher Ereignismakros, die sonst üblicherweise bei Benutzerinteraktion ausgeführt werden (z.B. nach Speichern, Löschen, Einfügen eines Datensatzes, ...).
ACCFLAG_NO_OBJECTLOAD	Diese Option sorgt dafür, dass nach einer Aktion des Datenzugriffsobjektes die von ihm abhängigen Eingabeobjekte nicht erneut gefüllt werden. Diese bleiben unverändert.

Der Funktionswert ist vom Typ *integer*

Wird über die Modus-Steuerung eine Abfrage durchgeführt, so ist der Funktionswert oben beschrieben.

Ansonsten:

IDOK	Die Funktion wurde fehlerfrei ausgeführt.
IDCANCEL	Die Funktion konnte nicht ausgeführt werden.

Form_DuplicateObjects

HWND	hwndForm
long	lParam
int	flags

Diese Funktion dupliziert alle selektierten Objekte analog zur Funktion “Objekte duplizieren” des Formulareditors. Ein Dialog wird eröffnet, um die notwendigen Benutzereingaben (Position, Anzahl der Kopien, etc) entgegenzunehmen.

Es werden alle selektierten Objekte dupliziert, ist kein Objekt selektiert, so ist die Funktion ohne Wirkung

lParam ist für spätere Zwecke reserviert und muss 0 sein.

flags ist eine Bitliste aus folgenden möglichen Werten:

FM_BEEP	Im Fehlerfall einen Klang ausgeben
FM_UNDO	Ein UnDo-Item absetzen
FM_EDITMODE	Funktion ablehnen, falls nicht der Bearbeitungsmodus eingeschaltet ist

Der Funktionswert ist vom Typ *integer*.

IDOK	Die Funktion wurde fehlerfrei ausgeführt.
IDCANCEL	Der Dialog wurde abgebrochen oder es ist ein Fehler aufgetreten.

Form_EditCurrentObjectsFont

HWND hwndForm

Diese Funktion öffnet den Dialog zum Auswählen einer Schriftart und erlaubt das Ändern der Schrift des augenblicklich aktiven Eingabeobjekts. Wenn kein Eingabeobjekt aktiv ist (z.B. wenn sich das Formular im Editiermodus befindet), oder das Objekt über keine Schriftart verfügt (z.B. wenn es sich um ein Ankreuzfeld handelt), ist der Aufruf der Funktion ohne Wirkung.

Der Funktionswert ist vom Typ *integer*

IDOK Der Dialog wurde mit OK bestätigt.
IDCANCEL Der Dialog wurde abgebrochen oder gar nicht erst eröffnet.

Form_EditFormProperties

HWND hwndForm
int Action
long lParam
int Flags

Diese Funktion öffnet einen Dialog zum Bearbeiten von Formularparametern. Je nachdem, welche Parameter ediert werden sollen, werden unterschiedliche Dialog-Templates angezeigt.

hwndForm Handle des Formulars, dessen Parameter ediert werden sollen.

Action Angabe, was ediert werden soll. Folgende Werte können angegeben werden:

FORM_EDFOP_MACROS	Makros
FORM_EDFOP_PASSWORD	Passwort
FORM_EDFOP_GRID	Gitternetz
FORM_EDFOP_RASTER	Raster
FORM_EDFOP_CONFIG	Konfigurationsparameter
FORM_EDFOP_PRINT_PARAMS	Druckereinstellungen
FORM_EDFOP_COPYLIST	Ausfertigungen
FORM_EDFOP_PRINT_EXPIRATION	Verfallsdatum für Drucken

FORM_EDFOP_DISPLAY_OPTIONS	Anzeigeeinstellungen
FORM_EDFOP_OPTIONS	Sonstige Einstellungen (Optionen)
FORM_EDFOP_HTML_EXPORT	HTML Export Parameter
FORM_EDFOP_PDF_EXPORT	PDF Export Parameter
FORM_EDFOP_RUNTIME_OPTIONS	Laufzeiteinstellungen
FORM_EDFOP_WATERMARK	Wasserzeichen

lParam Reserviert für spätere Zwecke. Muss 0 sein.

Flags Bitliste aus einem oder mehreren der folgenden Werte:

FM_BEEP	Im Fehlerfall einen Klang ausgeben
FM_UNDO	Ein UnDo-Item absetzen
FM_EDITMODE	Funktion ablehnen, falls nicht der Bearbeitungs-modus eingeschaltet ist

Der Funktionswert ist vom Typ *integer*

IDOK Der Dialog wurde mit OK bestätigt.
IDCANCEL Der Dialog wurde abgebrochen.

Diese Funktion ersetzt folgende ältere Funktionen

Form_EditFormMacros
Form_EditPasswords
Form_EditGrid
Form_EditRaster
Form_EditConfig
Form_EditPrintParams
Form_EditCopyList
Form_EditPrintExpiration
Form_EditDisplayOptions
Form_EditOptions
Form_EditHtmlExport
Form_EditPdfExport

Die älteren Funktionen sollten nicht mehr verwendet werden.

Form_EditGlobalPrintOffsets

HWND hwndForm

Diese Funktion öffnet einen Dialog, welcher das Einstellen des horizontalen und vertikalen globalen Druckoffsets ermöglicht. Die Druckoffsets werden beim Drucken auf alle Objekte aufaddiert. Sie können positiv oder negativ sein. Es können also alle Objekte gemeinsam nach oben, unten, rechts oder links verschoben werden.

Mit den Funktionen *Form_SetGlobalPrintOffsetX* und *Form_SetGlobalPrint OffsetY* können die Offsets auch direkt – ohne einen Dialog zu öffnen – gesetzt werden.

Der Funktionswert ist vom Typ *integer*

TRUE Der Dialog wurde mit OK bestätigt.
FALSE Der Dialog wurde abgebrochen.

Verwandte Funktionen

Form_SetGlobalPrintOffsetX
Form_SetGlobalPrintOffsetY

Form_EditMasterReferenceObject

```
HWND   hwndForm
long   ObjectType
long   Flags
```

Diese Funktion öffnet einen Dialog zur Konfiguration der Grundeinstellungen, welche bei Neuanlage eines Objektes vorgeschlagen werden. Abhängig von *ObjectType* werden die entsprechenden Parameter für Text-, Eingabe-, Auswahl- und Ankreuzobjekte eingestellt. Flags ist für zukünftige Zwecke gedacht und muss mit Null vorbesetzt werden.

Gültige Angaben für *ObjectType* sind:

FOTYPE_TEXT	Öffnet den Dialog für Textobjekte.
FOTYPE_EDIT	Öffnet den Dialog für Eingabefelder.
FOTYPE_COMBO	Öffnet den Dialog für Auswahlfelder.
FOTYPE_CHECKBOX	Öffnet den Dialog für Ankreuzfelder.

Der Funktionswert ist vom Typ *integer*

IDOK	Der Dialog wurde mit OK bestätigt.
IDCANCEL	Der Dialog wurde abgebrochen.

Form_EditObjects

HWND	hwndForm
int	action
const char *	pageName
const char *	objName
int	objTypes
int	flags
long	lParam

Diese Funktion ermöglich das edieren verschiedener Objekteigenschaften mittels eines spezifischen Dialogs.

hwndForm bezeichnet den Handle des Formulars.

Action Was soll bearbeitet werden:

FORM_EDIT_FONT	Schriftart bearbeiten
FORM_EDIT_BGRCOLOR	Hintergrundfarbe bearbeiten
FORM_EDIT_FONTCOLOR	Schriftfarbe bearbeiten
FORM_EDIT_FRAMECOLOR	Rahmenfarbe bearbeiten
FORM_EDIT_COMMPROP	Gemeinsame Eigenschaften bearbeiten
FORM_EDIT_OUTLINE	Position und Größe bearbeiten
FORM_EDIT_SIZE	Größe bearbeiten

pageName == NULL Operation nur für die aktuelle Seite durchführen.

pageName != NULL Operation für alle Seiten mit dem Namen *pageName* durchführen. Wilde Namen (z.B. “*“ für alle Seiten) sind erlaubt.

objName == NULL Operation für Objekte mit beliebigen Namen durchführen.

objName != NULL Operation für alle Objekte mit dem Namen *objName* durchführen. Wilde Namen (z.B. “*“ für alle Objekte) sind erlaubt.

objTypes == 0	Operation für Objekte beliebigen Typs durchführen
objTypes == x	Bitliste aller Objekttypen, für die die Operation durchgeführt wird. Im Anhang OBJEKTTYPEN sind alle Objekttypen beschrieben.

flags ist eine Liste von Bits folgender Optionen:

FM_SELECTED	Operation nur für markierte Objekte ausführen
FM_INPUT	Operation nur für Eingabeobjekte ausführen
FM_GROUPED	Operation nur für gruppierte Objekte ausführen
FM_BEEP	Im Fehlerfall einen Klang ausgeben
FM_UNDO	Ein UnDo-Item absetzen
FM_EDITMODE	Funktion ablehnen, falls nicht der Bearbeitungsmodus eingeschaltet ist
FM_REFRESH	Änderungen sofort anzeigen

lParam ist für spätere Zwecke reserviert und sollte immer 0 sein.

Der Funktionswert ist vom Typ *integer*

IDOK	Ok. Dialog bestätigt.
0	Dialog wurde abgebrochen oder es ist ein Fehler aufgetreten.

Diese Funktion ersetzt folgende ältere Funktionen

Form_EditFonts
Form_EditColors
Form_EditFrameColors
Form_EditFontColors
Form_EditObjectProperties
Form_EditOutlines
Form_EditSizes

Form_EditParameterList

char *	Parameters
char *	Contents
long	Size
char *	Title
int	Zoom
long	Flags

Diese Funktion eröffnet einen Dialog zum Ausfüllen einer Parameterliste. Dies wird über ein temporär erzeugtes Formular realisiert, welches im Dialogfenster zum Editieren dargestellt wird.

Eine Liste von Namen der Parameter wird durch TAB (\t) getrennt auf Parameters übergeben. Die initialen Werte der Parameter werden ebenso getrennt durch TAB (\t) auf *Contents* übergeben. Leere Angaben sind hier durchaus erlaubt. Jeder Eintrag in der Parameterliste muss mit dem Namen des Parameters beginnen und kann danach weitere Informationen getrennt durch einen Vertikal-TAB (\v) enthalten. Als erste Zusatzinfo kann ein Datentyp angegeben werden. Dieser ist klartextlich zu notieren. Mögliche Angaben sind: TEXT, DATE, TIME, INTEGER, FLOAT. Die zweite Angabe beschreibt die horizontale Ausrichtung. Hier sind die Werte LEFT und RIGHT zulässig. Als drittes kann eine Formatangabe folgen. Nähere Informationen zur Formatbeschreibung enthält eine gesonderte Dokumentation. Werden keine Zusatzinformationen hinter dem Parameternamen angegeben, so wird linksbündiger Text vorausgesetzt.

Beispiel einer Parameterliste:

Name\v*TEXT*\v*LEFT*\t*Geburtsdatum*\v*DATE*\v*LEFT*\v*%D1-%M3-%Y2*

Hier werden zwei Parameter deklariert. Der erste vom Typ Text, der zweite ist ein Datum, welches in der Form 3-Jan-1955 formatiert werden soll. Beide Parameter werden linksbündig angezeigt.

Wenn der Dialog vom Benutzer mit OK bestätigt wird, so wird auf dem Parameter *Contents* der eingegebene Inhalt gelistet. Leere Inhalte sind hier durchaus erlaubt. Es werden maximal *Size* Zeichen übertragen. Wird die Liste länger, so wird sie abgeschnitten.

Über den Parameter Flags sind einige Funktionseinstellungen steuerbar. Mögliche Angaben für Flags sind:

EPL_CENTER_DIALOG	Das Dialogfenster wird auf dem Bildschirm zentriert dargestellt.

Der Funktionswert ist vom Typ *integer*

IDOK	Der Dialog wurde mit OK bestätigt, *Contents* enthält die neuen Inhalte.
IDCANCEL	Der Dialog wurde abgebrochen, *Contents* wurde nicht verändert.

Form_EditPrimarySelectedObjectExt

HWND	hwndForm
int	Action

Diese Funktion eröffnet einen Dialog, welcher die Änderung der Eigenschaften des augenblicklich primär markierten Objekts ermöglicht. Sie kann nur im *Ediermodus* aufgerufen werden. Andernfalls ertönt ein akustisches Signal.

Je nach Typ des Objekts, welches primär markiert ist, enthält der Dialog Eingabemöglichkeiten, die auf die spezifischen Eigenarten des betreffenden Objekts zugeschnitten sind.

Wenn kein Objekt primär markiert ist, ist der Aufruf dieser Funktion ohne Wirkung.

Flags Bitliste mit einem oder mehrerer der folgenden Werte:

FM_BEEP	Im Fehlerfall einen Klang ausgeben
FM_UNDO	Ein UnDo-Item absetzen
FM_EDITMODE	Funktion ablehnen, falls nicht der Bearbeitungsmodus eingeschaltet ist

Der Funktionswert ist vom Typ *integer*

TRUE	Der Dialog wurde mit OK bestätigt.
FALSE	Der Dialog wurde abgebrochen.

Diese Funktion ersetzt folgende ältere Funktionen

Form_EditPrimarySelectedObject

Die älteren Funktionen sollten nicht mehr verwendet werden.

Form_EditVerifyMode

HWND	hwndForm

Diese Funktion eröffnet einen Dialog, welcher es dem Benutzer ermöglicht, auszuwählen, in welchen Situationen semantische und syntaktische Prüfungen der Eingaben im Formular durchgeführt werden sollen. Es kann beispielsweise differenziert werden, ob die Prüfung einer falschen Eingabe unmittelbar nach Verlassen des Eingabeobjekts, welche die falsche Eingabe enthält, stattfinden soll oder ob die Prüfung für alle Objekte gemeinsam beim Speichern oder Drucken des Formulars erfolgen soll.

Der Funktionswert ist vom Typ *integer*

TRUE	Der Dialog wurde mit OK bestätigt.
FALSE	Der Dialog wurde abgebrochen.

Form_Export

HWND	hwndForm
char *	fileName
int	fileType
long	flags

Diese Routine exportiert ein benanntes oder ein unbenanntes Formular in einem vorgebbaren Format in die Datei, deren Name als Parameter *fileName* und deren Dateityp als Parameter *fileTyp* angegeben wird.

Folgende Dateitypen können als Ziel beim Speichern verwendet werden:

FORM_FILETYPE_FVL	Standard-Formulardatei
FORM_FILETYPE_MFF	Metaformat für Formulare
FORM_FILETYPE_HTML	HTML-Datei
FORM_FILETYPE_PDF	PDF-Datei
FORM_FILETYPE_AUTOBYEXTENSION	Der Dateityp wird aus der Extension der Datei abgeleitet

Die Titeldatei bleibt unverändert. Das Change-Flag wird im Gegensatz zu der Funktion *Form_SaveAs* nicht zurückgesetzt.

Der Funktionswert ist vom Typ *integer*

0 Das Formular wurde fehlerfrei exportiert.

-1 Es trat ein Fehler auf. Eine Meldung wurde bereits angezeigt. Das Formular wurde nicht oder nicht vollständig exportiert.

Verwandte Funktionen

Form_SaveAs
Form_Save
Form_GetTitleFileName
Form_SetChangeFlag

Form_FindNextObject

HWND	hwndForm
HFOB	refObject
const char *	pageName
const char *	ObjName
long	ObjTypes

Diese Funktion ermittelt den Handle eines Objektes. Das Objekt wird auf Basis verschiedener Parameter im Formular gesucht.

refObject == NULL	Suche das erste Objekt mit den angegebenen Eigenschaften.
refObject != NULL	Suche das nächste Objekt mit den angegebenen Eigenschaften nach dem angegebenen

pageName == "watermark"	Objekte im Wasserzeichen finden.
pageName == NULL	Objekte auf der aktuellen Seite finden.
pageName != NULL	Objekte auf allen Seiten mit dem Namen *pageName* finden. Wilde Namen (z.B. "*" für alle Seiten) sind erlaubt.

objName == NULL	Objekte mit beliebigen Namen finden.
objName != NULL	Nur Objekte mit dem Namen *objName* finden. Wilde Namen (z.B. "*" für alle Objekte) sind erlaubt.

objTypes == 0	Objekte beliebigen Typs finden
objTypes == x	Bitlsite aller Objekttypen, die gefunden werden sollen. Im Anhang OBJEKTTYPEN sind alle Objekttypen beschrieben.

Funktionswert vom Typ HFOB

Wert != 0	Der Handle des nächsten gefundenen Objektes.
Wert == 0	Es wurde kein (weiteres) Objekt gefunden.

Beispiele

Die folgende Schleife setzt den Inhalt aller Objekte eines Formulars zurückt:

```
HFOB Objekt = (HFOB) 0;

while ((Object = Form_FindNextObject (hForm, Object, "*", "*", FOTYPE_ALL)) != (HFOB) 0)
   FormObj_SetText (Object, "", NOTIFY_OFF, MACROCALL_OFF);
```

Die folgende Schleife besetzt den Inhalt aller Eingabeobjekte auf allen Seiten, deren Name mit T beginnt, mit dem Text Hallo vor:

```
HFOB Objekt = (HFOB) 0;

while ((Object = Form_FindNextObject (hForm, Object, "T*", "*", FOTYPE_INPUT)) != (HFOB) 0)
   FormObj_SetText (Object, "Hallo", NOTIFY_OFF, MACROCALL_OFF);
```

Die folgende Sequenz ermittelt den aktuellen Inhalt des Eingabefeldes ***Klaus***:

```
HFOB Objekt = (HFOB) 0;
char Puffer [100];

Object = Form_FindNextObject (hForm, Object, "*", "Klaus", FOTYPE_EDIT);
FormObj_GetText (Object, Puffer, sizeof (Puffer));
```

Verwandte Funktionen

Form_GetCurrentInputObject

Form_GetChangeFlag

HWND hwndForm

Diese Funktion liefert zurück, ob ein Formular seit dem Öffnen oder Anlegen verändert worden ist. Im Ausfüllmodus gilt ein Formular als verändert, wenn der Benutzer den Inhalt irgendeines Eingabeobjekts verändert hat. Im Ediermodus gilt ein Formular als verändert, wenn die Struktur verändert wurde. Hierzu gehört das Anlegen oder Löschen von Objekten, das Vergrößern, Verschieben, etc.

Unabhängig davon, ob ein Formular wirklich verändert wurde, kann die interne Merkvariable mittels der Funktion *Form_SetChangeFlag* jederzeit auf verändert oder auf nicht verändert gesetzt werden.

Der Funktionswert ist vom Typ *integer*

1 Das Formular wurde verändert.
0 Das Formular wurde nicht verändert.

Verwandte Funktionen
Form_SetChangeFlag

Form_GetColorMode

HWND hwndForm

Diese Funktion liefert zurück, ob ein Formular im Schwarz/Weiß-Modus betrieben wird oder in Normaldarstellung. Die Funktion kann sowohl im Ediermodus als auch im Ausfüllmodus jederzeit aufgerufen werden.

Wenn sich ein Formular im Schwarz/Weiß-Modus befindet, werden die Hintergründe aller Objekte weiß und die Ränder und Schriften aller Objekte schwarz dargestellt.

Der Schwarz/Weiß-Modus kann jederzeit mittels der Funktion Form_SetColorMode umgeschaltet werden.

Der Funktionswert ist vom Typ *integer*

COLORMODE_NORMAL	Normaldarstellung
COLORMODE_BW	Schwarz/Weiß-Darstellung

Verwandte Funktionen
Form_SetColorMode

Form_GetColorProperty

HWND	hwndForm
long	property

Diese Funktion liefert einen der Farbwerte des durch *hwndForm* spezifizierten Formulars zurück.

property definiert, welche Farbe gelesen werden soll.
Folgende Werte sind möglich:

FORM_PROP_COLOR_WINDOW_BGR	Farbe des Fensters der Anwendung
FORM_PROP_COLOR_PAGE_SHADOW	Farbe des Seiten-Schattens
FORM_PROP_COLOR_ACTIVE_INPUT	Farbe, in der das aktive Eingabefeld dargestellt wird. Die Farbe wird nur verwendet, wenn die Eigenschaft FORM_PROP_ACTIVE_INPUT_DISPLAY mittels der Routine *Form_SetLongProperty* auf den Wert 1 gesetzt wird.
FORM_PROP_COLOR_COMMENT_AVAIL	Farbe, in der Eingabeobjekte dargestellt werden, für die ein Kommentar verfügbar ist
FORM_PROP_COLOR_RASTER	Farbe, in der das Gitternetz angezeigt wird, wenn es eingeschaltet ist

Alle oben aufgeführten Eigenschaften können auch mittels der Funktion *Form_SetColorProperty* gesetzt werden.

Der Funktionswert ist vom Typ COLORREF.

Der aktuell eingestellte Wert für die angefragte Farbe wird zurückgeliefert.

Verwandte Funktionen

Form_GetXmlPropertyByAttrName
Form_GetLongProperty
Form_GetXmlProperty
Form_SetColorProperty

Form_GetCurrentInputObject

HWND hwndForm

Diese Funktion liefert den Handle des augenblicklich aktiven Eingabeobjekts zurück. Der Handle hat den Datentyp HFOB. Es stehen eine Reihe von Funktionen zur Verfügung, mit denen die Eigenschaften von Objekten abgefragt oder verändert werden können. Diese Funktionen erwarten den Handle des betroffenen Objekts als Eingabeparameter.

Wenn sich das Formular im Ediermodus befindet, wird immer der Wert 0 zurückgeliefert, da im Ediermodus kein aktives Eingabeobjekt existiert.

Der Funktionswert ist vom Typ HFOB (Objekt-Handle)

hFob Handle des augenblicklich aktiven Eingabeobjekts.
0 Es ist momentan kein Eingabeobjekt aktiv.

Verwandte Funktionen

Form_GetPrimarySelectedObject
FormObj_GetXmlProperty

Form_GetCurrentInputObjectName

HWND hwndForm
char * Name
int maxLngName

Diese Funktion liefert den Namen des augenblicklich aktiven Eingabeobjekts auf dem Parameter Name zurück. Es werden maximal m*axLngName* Zeichen übertragen. Das Ergebnis ist – auch wenn es abgeschnitten wurde – in jedem Falle mit einem Nullbyte terminiert.

Wenn das Objekt keinen Namen hat, wird ein Leerstring zurückgeliefert. Wenn momentan kein Eingabeobjekt aktiv ist (z.B: im Ediermodus), ist der Inhalt des Ergebnispuffers Name undefiniert.

Der Funktionswert ist vom Typ *integer*

0 Der Name wurde auf dem Parameter Name abgelegt
-1 Es ist momentan kein Eingabeobjekt aktiv. Name ist ungültig.

Form_GetCurrentPageBgrMode

HWND hwndForm

Diese Funktion liefert den eingestellten Hintergrundmodus der aktuellen Seite des durch hwndForm spezifizierten Formulars zurück.

Der Funktionswert ist vom Typ *integer*

-1	Es gibt keine aktuelle Seite.
PAGEBGR_NONE	Die Seite hat keinen Hintergrund.
PAGEBGR_SOLID	Die Seite hat eine feste Hintergrundfarbe.
PAGEBGR_BITMAP	Die Seite hat ein eingeblendetes Bild als Hintergrund.

Verwandte Funktionen
Form_SetCurrentPageBgrMode

Form_GetEditOrderMode

HWND hwndForm

Diese Funktion liefert zurück, ob der Modus Festlegen der Eingabereihenfolge ein- oder ausgeschaltet ist. Für den Fall, dass er eingeschaltet ist, liefert die Funktion darüberhinaus die Information, welches Objekt als nächstes Eingabeobjekt zu bestimmen ist.

Der Funktionswert ist vom Typ *integer*

n > 0 Der Modus Eingabereihenfolge festlegen ist aktiv. Das nächste Objekt, dessen Eingabeindex festgelegt wird, ist das Objekt mit der Nummer n. Die Numerierung zählt von 1 an.

0 Der Modus Eingabereihenfolge festlegen ist nicht aktiv.

-1 Es ist ein Fehler aufgetreten. Eine entsprechende Fehlermeldung wurde bereits angezeigt.

Verwandte Funktionen
Form_SetEditOrderMode

Form_GetHelpFileName

char * *Name*
int *NameSize*

Diese Funktion liefert den Namen der Hilfedatei der Formular-DLL zurück.

Der Parameter *Name* zeigt auf einen Puffer, auf dem der Name der Hilfedatei zurückgeliefert wird. Es werden nicht mehr als *NameSize* Zeichen übertragen. Das Ergebnis wird mit einem Nullbyte abgeschlossen – auch wenn es abgeschnitten wird.

Diese Funktion liefert keinen Wert zurück.

Verwandte Funktionen
Form_GetMacroHelpFileName

Form_GetInstance

Diese Funktion hat keine Parameter

Diese Funktion liefert den *Instance-Handle* der Formular-DLL zurück. Dieser kann bei späteren Windows API-Aufrufen, z.B. Auslesen von Resourcen der Formularverwaltung, mitgegeben werden.

Der Funktionswert ist vom Typ HINSTANCE

Form_GetLongProperty

HWND hwndForm
longProperty

Diese Funktion liest die numerische Eigenschaft *Property* des durch *hwndForm* spezifizierten Formulars und liefert den Wert als *integer* Funktionswert zurück.

Folgende numerische Eigenschaften können z.Zt. abgefragt werden:

FORM_PRINTPROP_BLACKWHITE	Gibt die Eigenschaft ‚Druck in Schwarz/Weiß' zurück. Ist der Wert != 0, wird das Formular monochrom ausgedruckt.
FORM_PRINTPROP_COPIES	Gibt die Eingenschaft der ‚Anzahl der zu druckenden Kopieen' zurück.
FORM_PRINTPROP_COPYLISTMODE	Gibt die Eigenschaft ‚Ausfertigungen drucken' zurück. Ist der Wert != 0, werden alle konfigurierten Ausfertigungen gedruckt, anderenfalls nicht.
FORM_PRINTPROP_DUPLEX	Gibt die Eigenschaft des ‚Doppelseitigen Drucks' zurück.
FORM_PRINTPROP_FROMPAGE	Gibt den Wert der ersten zu druckenden Seite zurück, wenn die Eigenschaft FORM_PRINTPROP_PAGEMODE auf den Wert PAGEMODE_RANGE gesetzt ist.

FORM_PRINTPROP_ONLYCONTENT	Gibt die Eigenschaft ‚Vordruck ausblenden' zurück. Ist der Wert != 0, wird nur der vom Benutzer eingegebene Inhalt des Formulars gedruckt.
FORM_PRINTPROP_PAGEMODE	Gibt die Eigenschaft der zu druckenden Seiten zurück. Mögliche Angaben sind: PAGEMODE_ALL: es werden alle Seiten gedruckt PAGEMODE_RANGE: es wird ein Seitenbereich grdruckt PAGEMODE_CURRENT: es wird nur die aktuelle Seite gedruckt
FORM_PRINTPROP_TOPAGE	Gibt den Wert der letzten zu druckenden Seite zurück, wenn die Eigenschaft FORM_PRINTPROP_PAGEMODE auf den Wert PAGEMODE_RANGE gesetzt ist.
FORM_PRINTPROP_XOFFSET FORM_PRINTPROP_YOFFSET	Gibt die Eigenschaften des horizontalen bzw. vertikalen Druckoffsets zurück. Diese sind nicht zu verwechseln mit den global (im Editor oder Filler) eingestellten, über alle Formulare hinweg gültigen Offsets. Die hier eingestellten Werte werden formularspezifisch zu den globalen Werten hinzu addiert.
FORM_PRINTPROP_ZOOM	Gibt die Eigenschaft ‚Vergrößerung' in Prozent zurück.
FORM_PROP_ACTIVE_INPUT_DISPLAY	1 = aktives Eingabefeld hervorheben 0 = aktives Eingabefeld nicht hervorheben
FORM_PROP_AUTOZOOM_MODE	Autozoom-Modus FORM_AUTOZOOM_NONE kein automatisches Zoomen FORM_AUTOZOOM_WIDTH Zoomen auf Fensterbreite FORM_AUTOZOOM_HEIGHT Zoomen auf Fensterhöhe

FORM_PROP_COLORMODE	0 Darstellung in Farbe COLORMODE_BW Darstellung Schwarz-Weiss
FORM_PROP_CURRENT_PAGE_NUMBER	Nummer der aktuellen Seite (Zählung ab 1)
FORM_PROP_DISABLE_WARNING_BOXES	Abschalten von Warnungen
FORM_PROP_DISABLED_COMPONENTS	Bitliste, die einzelne Bedienungs-komponenten der Software ausschaltet. FORMCOMP_BUTTON_MACRO Abschalten der Makroeingabe FORMCOMP_BUTTON_HTMLJAVA Abschalten der Eingabe von Javaskript für HTML-Export FORMCOMP_BUTTON_PDFJAVA Abschalten der Eingabe von Javaskript für PDF-Export FORMCOMP_DATA_ACCESS Abschalten der Datenzugriffsobjekte
FORM_PROP_DISPDIST_BOTTOM	Unterer Rand zwischen Fenster und Formularseite in Pixeln
FORM_PROP_DISPDIST_HORI	Seitlicher Rand zwischen Fenster und Formularseite in Pixeln
FORM_PROP_DISPDIST_INTERPAGE	Abstand zwischen zwei Formularseiten in der Anzeige in Pixeln
FORM_PROP_DISPDIST_TOP	Oberer Rand zwischen Fenster und Formularseite in Pixeln
FORM_PROP_GRID	Rasterabstand in Mikrometern

FORM_PROP_HELP_MODE	Hilfe-Modus FORM_HELPMODE_DISABLED Die Hilfefunktionalität ist ausgeschaltet. FORM_HELPMODE_STANDARD Die Standard-Hilfe der Windows-Umgebung wird verwendet. Diese Option ist für Fremdapplikationen gedacht, welche eine eigene Hilfe für das Gesamtprodukt implementieren.
FORM_PROP_MODAL_FLAGS	Bitliste mit Optionen für den Betrieb als modaler Dialog: FORM_MODALFLAG_MAXIMIZE Dialog maximieren
FORM_PROP_MODAL_HEIGHT	Höhe in Pixeln bei Verwendung des Formulars als modaler Dialog
FORM_PROP_MODAL_WIDTH	Breite in Pixeln bei Verwendung des Formulars als modaler Dialog
FORM_PROP_NEXTGROUPID	Nächste zu vergebende ID zur Kennzeichnung einer Gruppe von Objekten
FORM_PROP_NO_SAVE_QUESTION	1 = Frage *Speichern* unterdrücken 0 = Frage *Speichern* nicht unterdrücken
FORM_PROP_PAGEKEY_MODE	Blättern-Modus Bestimmt das Verhalten der Tasten *Bild auf* und *Bild ab:* FORM_PAGEKEY_PAGING Blättern auf der Formularseite FORM_PAGEKEY_DBACCESS Blättern durch Datensätze beim Datenbankzugriff
FORM_PROP_RASTER_DISTANCE	Gitternetz-Abstand oder 0 (kein Gitternetz)

FORM_PROP_RASTER_ZORDER	ZORDER_BACKGROUND Gitternetz im Hintergrund anzeigen ZORDER_FOREGROUND Gitternetz im Vordergrund anzeigen
FORM_PROP_READONLY_MODE	Formular arbeitet im *Read Only* Modus
FORM_PROP_SCROLL_OFFSET_HORI	Horizontaler Scroll-Offset in Pixeln
FORM_PROP_SCROLL_OFFSET_VERT	Vertikaler Scroll-Offset in Pixeln
FORM_PROP_TOTAL_PAGES	Anzahl der Seiten des Formulars

Alle oben aufgeführten Eigenschaften mit Ausnahme von

FORM_PROP_TOTAL_PAGES
FORM_PROP_CURRENT_PAGE_NUMBER
FORM_PROP_HORI_SCROLL_OFFSET
FORM_PROP_VERT_SCROLL_OFFSET

können auch mittels der Funktion *Form_SetLongProperty* gesetzt werden.

Der Funktionswert ist vom Typ long.

Der aktuell eingestellte Wert für die angefragte Eigenschaft wird zurückgeliefert.

Verwandte Funktionen

Form_GetXmlPropertyByAttrName
Form_SetLongProperty
Form_GetXmlProperty
Form_GetColorProperty

Form_GetMacroHelpFileName

char * *FileName*
int *MaxLng*

Diese Funktion liefert den Namen der Makro-Hilfedatei zurück. Der Name wird als Zeichenfolge auf der Variablen *FileName* abgelegt. Es werden nicht mehr als *MaxLng* Zeichen übertragen. Das Ergebnis wird in jedem Fall mit einem Nullbyte abgeschlossen. Dies gilt auch dann, wenn es abgeschnitten wurde, weil der Puffer zu kurz ist.

Die Funktion liefert keinen Funktionswert zurück.

Verwandte Funktionen

Form_PerformMacroHelp
Form_GetHelpFileName

Form_GetPageHandle

HWND	hwndForm
int	number

Diese Funktion liefert den Handle der Seite (HFPAGE) mit der Nummer *number* zurück. Die Seiten werden von 1 an gezählt.

Der Funktionswert ist vom Typ HFPAGE (Seiten-Handle)

hFPage != 0	Handle der Seite mit der Nummer *number*.
hFPage == 0	Es wurde eine ungültige Seitennummer übergeben.

Form_GetPointedObject

HWND	hwndForm
POINT *	Point
long	objTypes

Diese Funktion liefert den Handle des Objektes zurück, dessen Umriss die Koordinaten des Parameters Point beinhaltet. Die Koordinaten müssen in Pixel relativ zum Fenster des mit *hwndForm* spezifizierten Formulars angegeben werden.

Diese Werte werden z.B. mit den Host-Window-Nachrichten FORMMSG_LBUTTONDOWN und FORMMSG_LBUTTONUP übergeben.

objTypes *b*estimmt, ob alle oder nur eine Auswahl an Objekttypen erkannt werden sollen. Alle Objekttypen sind im Anhang OBJECTTYPEN erläutert.

Der Funktionswert ist vom Typ HFOB (Objekt-Handle).

hFob Handle des Objekts, auf welches die angegebenen Koordinaten zeigen.
0 Die Koordinaten liegen außerhalb aller Objekte.

Verwandte Funktionen
Form_PixelToPos
FormObj_GetXmlProperty

Form_GetPrimarySelectedObject

HWND hwndForm

Diese Funktion liefert den Handle des augenblicklich primär markierten Objekts zurück. Der Handle hat den Datentyp HFOB. Es stehen eine Reihe von Funktionen zur Verfügung, mit denen die Eigenschaften von Objekten abgefragt oder verändert werden können. Diese Funktionen erwarten den Handle des betroffenen Objekts als Eingabeparameter.

Wenn sich das Formular im Ausfüllmodus befindet, wird immer der Handle 0 zurückgeliefert, da hier keine Objekte markiert werden können.

Der Funktionswert ist vom Typ HFOB (Objekt-Handle).

hFob Handle des augenblicklich primär markierten Objekts.
0 Es ist momentan kein Eingabeobjekt aktiv.

Verwandte Funktionen
Form_FindNextObject
Form_GetCurrentInputObject

Form_GetTitleFileName

```
HWND      hwndForm
char *    DateiName
longMaxLng
```

Diese Funktion liefert den Namen der Datei, aus der ein Formular eröffnet wurde, auf dem Parameter *DateiName* zurück. Dies ist der Dateiname, der im Fenstertitel angezeigt wird. Es werden maximal *MaxLng* Zeichen übertragen. Das Ergebnis wird in jedem Fall mit einem Nullbyte abgeschlossen. Dies gilt auch dann, wenn es abgeschnitten wurde, weil der Puffer zu kurz ist.

Wenn es sich um ein neu angelegtes Formular handelt, wird ein leerer String zurückgeliefert.

Die Funktion liefert keinen Funktionswert zurück.

Verwandte Funktionen
Form_MakeUntitled

Form_GetXmlPrintProperty

```
HWND          hwndForm
const char *  attr
char *        result
int           size
```

Diese Funktion liest eine Druck-Eigenschaft des Formulars *hwndForm.* Die Eigenschaft wird durch den Attribut-Namen *attr* spezifiziert. Der Wert der Eigenschaft wird unabhängig vom Typ der Eigenschaft immer als Zeichenkette auf dem Parameter *result* übergeben. Es werden nicht mehr als *size* Zeichen übertragen. Das Ergebnis ist immer nullterminiert.

Alle Druck-Eigenschaften sind im Anhang PRINT XML PROPERTIES beschrieben.

Der Funktionswert ist vom Typ *integer*

0	Die Eigenschaft wurde wie gewünscht zurückgeliefert.
-1	Die Eigenschaft *attr* ist unbekannt oder konnte nicht gelesen werden.

Verwandte Funktionen
Form_GetXmlPropertyByAttrName
Form_SetXmlPrintProperty

Form_GetXmlPropertyByAttrName

HWND	hwndForm
const char *	AttrName
char *	Result
int	Size

Diese Funktion liest eine Eigenschaft des Formulars *hwndForm.* Die Eigenschaft wird durch den Attribut-Namen *AttrName* spezifiziert. Der Wert der Eigenschaft wird unabhängig vom Typ der Eigenschaft immer als Zeichenkette auf dem Parameter *Result* übergeben. Es werden nicht mehr als *Size* Zeichen übertragen. Das Ergebnis ist immer nullterminiert.

Wenn die Eigenschaft numerisch (ganzzahlig) ist, wird der numerische Wert als String zurückgeliefert, z.B. "123".

Wenn die Eigenschaft eine Farbe ist, wird der Wert als String der Form R,G,B zurückgeliefert, wobei die drei Komponenten R, G und B die numerischen Werte der Farben *Rot, Grün* und *Blau* in dezimaler Darstellung repräsentieren. 255,255,000 repräsentiert beispielsweise die Farbe *Gelb.*

Alle Formular-Eigenschaften sind im Anhang FORM XML PROPERTIES beschrieben.

Alle Eigenschaften können mit Hilfe der Funktion *Form_SetXmlPropertyByAttrName* auch verändert werden.

Der Funktionswert ist vom Typ *integer*

0	Die Eigenschaft wurde wie gewünscht gesetzt.
-1	Die Eigenschaft *AttrName* ist unbekannt oder konnte nicht gesetzt werden.

Verwandte Funktionen

Form_GetLongProperty
Form_GetColorProperty
Form_SetXmlPropertyByAttrName
Form_SetXmlPrintProperty

Form_GetVersion

char * *Version*
int *MaxLng*

Die Funktion *Form_GetVersion* liefert die aktuelle Version der Formular-DLL zurück. Die Version wird als Zeichenfolge auf der Variablen *Version* abgelegt. Es werden nicht mehr als *MaxLng* Zeichen übertragen. Das Ergebnis wird in jedem Fall mit einem Nullbyte abgeschlossen. Dies gilt auch dann, wenn es abgeschnitten wurde, weil der Puffer zu kurz ist.

Die Funktion liefert keinen Funktionswert zurück.

Form_GetWorkMode

HWND hwndForm

Diese Funktion liefert zurück, ob sich ein Formular im Ausfüllmodus oder im Ediermodus befindet. Eine Reihe von Funktionen können nur im Ediermodus ausgeführt werden, da sie im Ausfüllmodus keinen Sinn ergeben.

Der Funktionswert ist vom Typ *integer.*

FORMMODE_EDIT Das Formular befindet sich im Ediermodus.
FORMMODE_RUN Das Formular befindet sich im Ausfüllmodus.

Verwandte Funktionen
Form_SetWorkMode

Form_GetZoomFactor

HWND hwndForm

Diese Funktion liefert den aktuellen Vergrößerungsfaktor des Formulars zurück.

Der Vergrößerungsfaktor kann mittels der Funktion *Form_EditZoomFactor* vom Benutzer per Dialog eingestellt oder mittels der Funktion *Form_SetZoomFactor* direkt gesetzt werden.

Der Funktionswert ist vom Typ *integer*.

Augenblicklich eingestellter Vergrößerungsfaktor in Prozent

Verwandte Funktionen
Form_SetZoomFactor
Form_EditZoomFactor

Form_GotoInputObject

HWND	hwndForm
int	Mode
long	Flags

Diese Funktion navigiert abhängig von der Angabe Mode innerhalb des Formulars hwndForm. Flags ist für zukünftige Zwecke gedacht und muss mit Null vorbesetzt werden.

Gültige Angaben für *Mode* sind:

FORM_GOTO_FIRST	Sprung auf das erste Eingabeobjekt des gesamten Formulars.
FORM_GOTO_PREV	Sprung auf das vorige Eingabeobjekt.
FORM_GOTO_NEXT	Sprung auf das nächste Eingabeobjekt.
FORM_GOTO_LAST	Sprung auf das letzte Eingabeobjekt des gesamten Formulars.

Die Funktion liefert keinen Funktionswert zurück.

Form_IsBgrClipboardAvailable

Diese Funktion prüft, ob sich ein Objekt-Hintergrund in der Zwischenablage befinden.

Der Funktionswert ist vom Typ *integer.*

1 Es ist ein Objekt-Hintergrund in der Zwischenablage
0 Es ist keine Objekt-Hintergrund in der Zwischenablage

Form_IsCurrentObjectMultilineEdit

HWND hwndForm

Diese Funktion prüft, ob das Formular im Ausfüllmodus ist und wenn ja, ob das augenblicklich aktive Eingabeobjekt ein Mehrzeiliges Eingabefeld ist.

Der Funktionswert ist vom Typ *integer.*

0 Das Formular ist nicht im Ausfüllmodus oder das augenblicklich aktive Eingabefeld ist kein mehrzeiliges Eingabefeld.

1 Das Formular ist im Ausfüllmodus und das augenblicklich aktive Eingabeobjekt ist ein mehrzeiliges Eingabefeld.

Form_IsFormClipboardAvailable

Diese Funktion prüft, ob sich formularspezifische Informationen in der Zwischenablage befinden. Diese sind z.B. Objekte, die im Ediermodus ausgeschnitten oder in die Zwischenablage kopiert wurden.

Der Funktionswert ist vom Typ *integer.*

1 Es ist eine formularspezifische Information in der Zwischenablage.
0 Es ist keine formularspezifische Information in der Zwischenablage.

Form_IsFormWindow

HWND hwndForm

Diese Funktion prüft, ob *hwndForm* ein gültiger Formularfenster-Handle ist. Beim Öffnen eines Formulars mittels der Funktion *Form_Open_Expanded* wird der Handle des geöffneten Formularfensters als Funktionswert zurückgeliefert.

Der Funktionswert ist vom Typ *integer.*

1 *hwndForm* ist ein gültiger Formularfenster-Handle.
0 *hwndForm* ist kein Formularfenster-Handle.

Form_LoadFromProfile

HWND hwndForm
char * DateiName
long ObjektTypen

Diese Funktion lädt den Inhalt aller benannten Objekte aus einer *Profile-Datei*. Der Name der *Profile-Datei* wird auf dem Parameter ***DateiName*** übergeben.

Der Parameter *ObjektTypen* bestimmt, welche Arten von Objekten gelesen werden sollen. Es können ein oder mehrere Objekttypen durch ein logisches oder miteinander kombiniert werden. Außerdem können folgende Objektgruppen angesprochen werden:

FOTYPE_ALL es werden alle Objekte behandelt
FOTYPE_INPUT es werden nur Eingabeobjekte behandelt

Es werden alle Objekte des angegebenen Typs geladen, deren Namen in der Sektion *[Objects]* der *Profile-Datei* als Einträge genannt sind. Alle anderen Objekte werden nicht verändert.

Der Inhalt von Objekten, die in der *Profile-Datei* genannt sind und denen kein Wert (also ein Leerstring) zugeordnet ist, wird gelöscht.

Der Funktionswert ist vom Typ *integer.*

0 Die Funktion wurde fehlerfrei ausgeführt.
-1 Es trat ein Fehler auf. Eine entsprechende Meldung wurde bereits angezeigt.

Verwandte Funktionen

Form_SaveToProfile
Form_LoadProfileExtended
Form_SaveProfileExtended
Form_LoadObjects
Form_SaveObjects

Form_LoadFromProfileExtended

HWND	hwndForm
char *	fileName
char *	section
long	objTypes
int	flags

Diese Funktion lädt den Inhalt aller benannten Objekte aus einer *Profile-Datei*. Der Name der *Profile-Datei* wird auf dem Parameter *fileName*, der Name der auszulesenden Sektion wird auf auf *section* übergeben.

Der Parameter *objTypes* bestimmt, welche Arten von Objekten gelesen werden sollen. Es können ein oder mehrere Objekttypen durch ein logisches oder miteinander kombiniert werden. Außerdem können folgende Objektgruppen angesprochen werden:

FOTYPE_ALL	es werden alle Objekte behandelt
FOTYPE_INPUT	es werden nur Eingabeobjekte behandelt

Die verfügbaren Objekttypen sind im gleichnamigen Anhang OBJECTTYPEN beschrieben.

Es werden alle Objekte geladen, deren Namen in der Sektion der Profile-Datei als Einträge genannt sind. Alle anderen Objekte werden nicht verändert.

Der Inhalt von Objekten, die in der Profile-Datei genannt sind und denen kein Wert (also ein Leerstring) zugeordnet ist, wird gelöscht.

Flags ist eine Bitliste der folgenden Werte:

FORM_PROFILE_NEUTRAL_DATA	Die Daten werden im Profile unformatiert erwartet
FORM_PROFILE_FORMATTED_DATA	Die Daten werden im Profile formatiert erwartet
FORM_PROFILE_SILENT_ERROR	Fehler werden nicht angezeigt, sondern ignoriert

Der Funktionswert ist vom Typ *integer.*

0 Die Funktion wurde fehlerfrei ausgeführt.
-1 Es trat ein Fehler auf. Eine entsprechende Meldung wurde bereits angezeigt.

Verwandte Funktionen
Form_SaveToProfileExtended
Form_LoadObjects
Form_SaveObjects

Form_MakeUntitled

HWND hwndForm

Diese Funktion setzt ein Formular, welches aus einer Datei geöffnet wurde oder auf andere Weise mit einer Titel-Datei assoziiert ist, in den Zustand Ohne Namen.

Der Funktionswert ist vom Typ *integer.*

0 Die Funktion wurde fehlerfrei ausgeführt.
-1 Es trat ein Fehler auf. Eine entsprechende Meldung wurde bereits angezeigt.

Verwandte Funktionen
Form_GetTitleFileName

Form_MuToString

HWND	hwndForm
long	Wert
int	EinheitKategorie
char *	Ergebnis
int	MaxLngErgebnis

Diese Funktion konvertiert eine Maßzahl, welche in Mikrometern angegeben ist, in eine Dezimalzahl-String. Dabei wird das für das Formular eingestellte Maßeinheitensystem (metrisch oder englisch) berücksichtigt. Ein eventuell notwendiges Dezimalkomma wird ebenfalls den landesspezifischen Einstellungen des Formulars entnommen.

Sowohl das Dezimalkomma als auch das Maßeinheitensystem können vom Benutzer mittels der Funktion *Form_EditConfig* eingestellt werden.

Die zu konvertierende Maßzahl wird in Mikrometern als ganze Zahl auf dem Parameter Wert übergeben.

Der Parameter *EinheitKategorie* legt fest, ob die Ausgabe in einer kleinen Einheit, z.B. in Millimetern oder in einer gröberen Einheit, z.B. Zentimetern erfolgen soll. Die Einheit hängt von diesem Parameter und vom im Formular eingestellten Maßeinheitensystem ab:

UNITCAT_SMALL	Die Konvertierung erfolgt in Millimeter, falls das metrische Einheitensystem eingestellt ist oder in Zoll, falls das englische Einheitensystem eingestellt ist.
UNITCAT_MED	Die Konvertierung erfolgt in Zentimeter, falls das metrische Einheitensystem eingestellt ist oder in Zoll, falls das englische Einheitensystem eingestellt ist.

Die fertige Dezimalzahl wird als String auf dem Parameter Ergebnis abgelegt. Es werden nicht mehr als *MaxLngErgebnis* Zeichen übertragen. Das Ergebnis wird mit einem Nullbyte abgeschlossen. Dies gilt auch dann, wenn das Ergebnis zu lang ist und abgeschnitten wurde.

Die Funktion liefert keinen Funktionswert zurück.

Form_Open_Expanded

HWND	hwndHost
char *	FormFile
int	FileType
long	Flags

Diese Funktion eröffnet ein Formularfenster.

Das neue Formularfenster wird immer im Ediermodus eröffnet. Wenn das Formular nicht im Edier- sondern im Ausfüllmodus betrieben werden soll, so muss direkt nach dem Aufruf *Form_Open_Expanded* die Funktion *Form_SetWorkModeExt* aufgerufen und das Formular in den Ausfüllmodus umgeschaltet werden.

Der Parameter *hwndHost* bezeichnet den Fenster-Handle des Host-Fensters. Das Host-Fenster ist ein Fenster beliebigen Typs, welches die Rahmenapplikation für die Abwicklung der Formularverarbeitung zur Verfügung stellen muss. Das Host Fenster kann ein MDI-Fenster, ein SDI-Fenster oder ein modaler Dialog sein.

Der Parameter *FormFile* bezeichnet die Formulardatei, welche eröffnet werden soll. Soll ein neues Formular angelegt werden, wie dies z.B. im Formulareditor der Fall ist, wenn der Benutzer die Funktion Neu aufruft, muss hier der Wert NULL angegeben werden.

Der Parameter *FileType* bezeichnet den Typ der Datei. Es werden folgende Dateitypen unterstützt:

FORM_FILETYPE_FVL Standard-Formulardatei
FORM_FILETYPE_NIL Es wird keine Formulardatei angegeben

Soll ein neues Formular angelegt werden, wie dies z.B. im Formulareditor der Fall ist, wenn der Benutzer die Funktion *Neu* aufruft, muss hier der Wert FORM_FILETYPE_NIL angegeben werden.

Auf Parameter *Flags können* folgende Optionen angegeben werden:

FOEX_FLAG_CREATE_AUTO_HOST	Wenn dieses Flag angegeben wird, legt die Formularsoftware automatisch ein Host-Fenster an. In diesem Fall braucht kein Host-Fenster übergeben zu werden. Der Parameter *hwndHost* kann dann mit NULL besetzt werden. Ein automatisch angelegtes Host-Fenster wird beim Schließen des Formulars mittels *Form_Close* automatisch wieder geschlossen. Wenn dieses Flag gesetzt ist, **muss** das Formular daher mittels ***Form_Close*** geschlossen werden.
FOEX_FLAG_NO_LOCK	Die Formulardatei wir mit SH_DENYNO eröffnet und bleibt ungelockt.
FOEX_NO_OPEN_ERROR_MESSAGES	Wenn beim Öffnen des Formulars ein Fehler auftritt, werden keine Fehlermeldungen angezeigt.
FOEX_NO_PRINTER_ERROR_MESSAGES	Wenn beim Öffnen des Druckers oder beim Einlesen des Drucker-IC bzw. des Drucker-DC ein Fehler auftritt, wird dieser nicht angezeigt.

Der Funktionswert ist vom Typ HWND (Formular-Handle).

hwnd Es wurde ein Formularfenster als Unterfenster des angegebenen Host-Fensters eröffnet. Der zurückgelieferte Handle wird als Eingabeparameter in allen Funktionen verwendet, die sich auf dieses Formular beziehen sollen.

0 Es trat ein Fehler auf. Eine entsprechende Meldung wurde bereits angezeigt. Das Formularfenster ist nicht eröffnet worden.

Verwandte Funktionen

Form_Close

Form_ObjectApi

```
HWND          hwndForm
int           action
const char *  pageName
const char *  objName
int           objTypes
RECT *        rect
int           flags
long          lParam
```

Diese Funktion stellt verschiedene Operationen bereit, die für allen Objekten durchgeführt werden, die zu den übergebenen Parametern passen.

hwndForm bezeichnet den Handle des Formulars.

action Operation, die ausgeführt werden soll:

FORM_OAPI_COUNT	Objekte zählen
FORM_OAPI_REFRESH	Anzeige auffrischen
FORM_OAPI_DELETE	Objekte löschen
FORM_OAPI_COPY	Objekte in die Zwischenablage kopieren
FORM_OAPI_CUT	Objekte ausschneiden und in die Zwischenablage kopieren
FORM_OAPI_PASTE	Objekte aus der Zwischenablage einfügen
FORM_OAPI_TO_FOREGROUND	Objekte in den Vordergrund heben
FORM_OAPI_TO_BACKGROUND	Objekte in den Hintergrund heben
FORM_OAPI_GROUP	Objekte gruppieren
FORM_OAPI_UNGROUP	Objekte entgruppieren
FORM_OAPI_COPY_BGR	Hintergrund des primär markierten Objekts in die Zwischenablage kopieren
FORM_OAPI_PASTE_BGR	Hintergrund aus der Zwischenablage auf die Objekte übertragen

FORM_OAPI_COPY_TEXT	Text des primär markierten Objekts in die Zwischenablage kopieren
FORM_OAPI_PASTE_TEXT	Text aus der Zwischenablage in die Objekte übernehmen
FORM_OAPI_SET_DEFAULT	Die Voreinstellung übernehmen (für Eingabeobjekte)

pageName == NULL Operation nur für die aktuelle Seite durchführen.

pageName != NULL Operation für alle Seiten mit dem Namen *pageName* durchführen. Wilde Namen (z.B. "*" für alle Seiten) sind erlaubt.

objName == NULL Operation für Objekte mit beliebigen Namen durchführen.

objName != NULL Operation für alle Objekte mit dem Namen *objName* durchführen. Wilde Namen (z.B. "*" für alle Objekte) sind erlaubt.

objTypes == 0 Operation für Objekte beliebigen Typs durchführen

objTypes == x Bitliste aller Objekttypen, für die die Operation durchgeführt wird. Insbesondere sind folgende Bitkombinationen definiert:

FOTYPE_ALL	alle Objekttypen
FOTYPE_INPUT	Eingabeobjekte
FOTYPE_ACCESS	Datenzugriffsobjekte

Eine Liste aller Objekttypen ist im Anhang OBJEKTTYPEN definiert.

pageRect == NULL Operation unabhängig von der Position des Objekts durchführen.

pageRect != NULL Operation nur für Objekte durchführen, die vollständig innerhalb des übergebenen Rechtecks liegen (Seitenkoordinaten in Mikrometern).

flags ist eine Liste von Bits folgender Optionen:

FM_SELECTED	Operation nur für markierte Objekte ausführen
FM_INPUT	Operation nur für Eingabeobjekte ausführen

FM_GROUPED	Operation nur für gruppierte Objekte ausführen
FM_TABENTER	Operation nur für Eingabeobjekte ausführen, in die mit der TAB-Taste gesprungen werden kann
FM_VISIBLE	Operation nur für Objekte ausführen, die im Fenster sichtbar sind
FM_READONLY	Operation nur für schreibgeschützte Objekte ausführen
FM_BEEP	Im Fehlerfall einen Klang ausgeben
FM_UNDO	Ein UnDo-Item absetzen
FM_EDITMODE	Funktion ablehnen, falls nicht der Bearbeitungsmodus eingeschaltet ist
FM_CONFIRM	Der Anwender muss die Funktion vor dem Ausführen mittels eines Dialogs bestätigen
FM_REFRESH	Änderungen sofort anzeigen
FM_NOTIFY	Benachrichtigung über die Aktion an das Hostfenster senden

lParam ist für spätere Zwecke reserviert und sollte immer 0 sein.

Der Funktionswert ist vom Typ *integer*

n	Anzahl verarbeiteter Objekte
0	Es wurden keine Objekte verarbeitet

Diese Funktion ersetzt folgende ältere Funktionen

Form_ObjectsApi
Form_DeleteSelectedObjects
Form_CopySelectedObjects
Form_CutSelectedObjects
Form_SelectedObjectsToForground
Form_SelectedObjectsToBackground

Die älteren Funktionen sollten nicht mehr verwendet werden.

Form_PageApi

```
HWND    hwndForm
int     action
long    lParam
int     flags
```

Diese Funktion ermöglicht verschiedene Operationen für die Verwaltung der einzelnen Formularseiten.

hwndForm bezeichnet den Handle des Formulars.

action bezeichnet die Operation, die ausgeführt werden soll:

FORM_PAPI_EDIT_CURRENT	Öffnen des Dialogs zum bearbeiten der Parameter der aktuellen Seite
FORM_PAPI_INSERT_NEW	Neue Seite vor der augenblicklich aktiven Seite einfügen
FORM_PAPI_APPEND_NEW	Neue Seite anlegen und an das Formular anhängen
FORM_PAPI_DUPLICATE	Die augenblicklich aktive Seite duplizieren
FORM_PAPI_DELETE_CURRENT	Die augenblicklich aktive Seite löschen
FORM_PAPI_TOGGLE_BGRBMP	Auf er augenblicklich aktiven Seite zwischen Hintergrundbitmap und leerem Hintergrund umschalten
FORM_PAPI_SET_CURRENT	Festlegen, welches die augenblicklich aktive Seite ist. Parameter *lParam* benennt die Seite (zählweise ab 1)

lParam ist ein Parameter, dessen Bedeutung von *Action* abhängt (siehe Beschreibung der einzelnen Aktionen).

flags ist eine Liste folgender Werte:

FM_BEEP	Im Fehlerfall einen Klang ausgeben
FM_UNDO	Ein UnDo-Item absetzen
FM_EDITMODE	Funktion ablehnen, falls nicht der Bearbeitungsmodus eingeschaltet ist

Der Funktionswert ist vom Typ *integer*

IDOK	Der Dialog wurde mit OK bestätigt.
IDCANCEL	Der Dialog wurde abgebrochen.

Diese Funktion ersetzt folgende ältere Funktionen

Form_EditCurrentPage
Form_CreateNewPage
Form_DeleteCurrentPage

Die älteren Funktionen sollten nicht mehr verwendet werden.

Form_PerformBeginMacro

HWND hwndForm

Diese Funktion ruft das Makro auf, welches in der globalen Makroliste des Formulars unter der Bezeichnung bei Beginn der Bearbeitung abgelegt ist.

Die Funktion liefert keinen Funktionswert zurück.

Form_PerformMacroHelp

```
char *      Statement
```

Diese Funktion zeigt eine Onlinehilfe zu der auf Statement übergebenen Makrofunktion an. Die Routine wird üblicherweise von Makroeditoren (z.B. Liste aller Makrofunktionen im Formular-Editor) aufgerufen. Der Name der Hilfedatei für Makrofunktionen kann via *Form_GetMacroHelpFileName* ermittelt werden.

Die Funktion liefert keinen Funktionswert zurück.

Verwandte Funktionen

Form_GetMacroHelpFileName

Form_PixelToPos

```
HWND    hwndForm
long    xPix
long    yPix
long *  xMu
long *  yMu
```

Diese Funktion liefert den Handle der Seite (HFPAGE) zurück, welche die angegebenen Koordinaten beinhaltet. Die Eingabekoordinaten xPix und yPix sind relativ zum Formularfenster (*hwndForm*) in Pixeln anzugeben. Diese Werte werden z.B. von der Hostfensternachricht FORMMSG_LBUTTONDOWN mitgeliefert. Zeigt die Position auf eine Seite des Formulars, so werden die Koordinaten umgerechnet in Mikrometer und relativ zur Seite auf den Parametern *xMu* und *yMu* zurückgeliefert.

Der Funktionswert ist vom Typ HFPAGE (Seiten-Handle).

hFPage != 0	Handle der Seite, auf welches die angegebenen Koordinaten zeigen.
hFPage == 0	Die Koordinaten liegen außerhalb aller Seiten des Formulars.

Verwandte Funktionen

Form_GetPointedObject
Form_GetPageHandle

Form_PresetAutoFormDesign

AUTOFORMDESIGN * *Design*
int *Style*

Diese Funktion besetzt die Struktur *Design* vom Typ *AUTOFORMDESIGN* für einen nachvolgenden Aufruf von z.B. *Form_CreateAutoForm* vor. Mit *Style* wird angegeben, welches Aussehen das Datenblatt haben soll.

Style kann folgende werte annehmen:

FORM_DESIGN_STYLE_DATABASE	Das Design soll einer einfachen Datenbankoberfläche gleichen, keine besonderen Stilmittel werden verwendet.
FORM_DESIGN_STYLE_CLASSIC	Das Design soll einem ansprechendem klassischem Datenbankformular gleichen.

Die Funktion liefert keinen Funktionswert zurück.

Verwandte Funktionen
Form_CreateAutoForm
Form_CreateDatabaseForm

Form_Print

HWND hwndForm
long Flags

Diese Funktion druckt das angegebene Formular. Alle Druckparameter (Seitenbereich, etc.) werden den im Formular gespeicherten Werten entnommen. Wenn für den Druck des Formulars andere Werte verwendet werden sollen, müssen diese mittels der dafür vorgesehenen Funktionen zuvor modifiziert werden.

Auf Wunsch kann vor dem Druck der Dialog zur Einstellung der Druckparameter eröffnet werden zu diesem Zweck muss auf dem Parameter Flags der Wert PRINTFLAG_SHOWDIALOG angegeben werden.

Der Funktionswert ist vom Typ *integer*.

-1 Es ist ein Fehler aufgetreten. Eine Meldung wurde bereits angezeigt.
n Es wurden insgesamt n Ausfertigungen gedruckt.

Verwandte Funktionen
Form_SetPrinterBin
Form_SetPrinterNameAndPort
Form_SetPrinterMode

Form_ReplaceTexts

```
HWND   hwndForm
char *  pageName
char *  objName
long    objTypes
char *  reserved
char *  reserved
long    flags
```

Diese Routine führt Textersetzungen innerhalb von Objekten durch, analog zur Funktion *Texte ändern* des Formulareditors.

hwndForm bezeichnet den Handle des Formulars.

pageName == NULL	Ersetzungen nur auf der aktuelle Seite durchführen.
pageName != NULL	Ersetzungen auf allen Seiten mit dem Namen *pageName* durchführen. Wilde Namen (z.B. "*" für alle Seiten) sind erlaubt.
objName == NULL	Ersetzungen in Objekten mit beliebigen Namen durchführen.
objName != NULL	Ersetzungen nur in Objekten mit dem Namen *objName* durchführen. Wilde Namen (z.B. "*" für alle Objekte) sind erlaubt.
objTypes == 0	Ersetzungen in Objekten beliebigen Typs durchführen
objTypes == x	Ersetzungen nur in den hier als Bitliste angegebenen Objekttypen durchführen. Eine Liste aller Objekttypen ist im Anhang OBJEKTTYPEN definiert.

flags ist eine Liste von Bits folgender Optionen:

FM_SELECTED	Operation nur für markierte Objekte ausführen
FM_INPUT	Operation nur für Eingabeobjekte ausführen
FM_GROUPED	Operation nur für gruppierte Objekte ausführen
FM_BEEP	Im Fehlerfall einen Klang ausgeben
FM_UNDO	Ein UnDo-Item absetzen
FM_EDITMODE	Funktion ablehnen, falls nicht der Bearbeitungsmodus eingeschaltet ist
FM_REFRESH	Änderungen sofort anzeigen
FM_NOTIFY	Benachrichtigung über die Aktion an das Hostfenster senden

Der Funktionswert ist vom Typ integer.

0 Die Funktion wurde fehlerfrei ausgeführt.
-1 Der Dialog wurde abgebrochen oder es ist ein Fehler aufgetreten.

Form_RunDatabaseWizzard

HWND hwndForm

Diese Funktion startet den Assistenten für das Anlegen eines Datenbankzugriffsformulars. Diese Funktion ist analog zur Funktion "Assistent für neues Formular mit Datenbankzugriff" des Formulareditors. *hwndForm* muss zuvor mittels der Routine *Form_Open_Expanded* angelegt oder eröffnet worden sein.

Es wird eine Datenbank und eine Tabelle oder Abfrage ausgewählt. Die im Formular anzulegenden Felder sowie die Sortierung der Datensätze können festgelegt werden. Am Ende wird der Stil des Datenblattes angefragt. Anschließend werden im Formular alle notwendigen Komponenten angelegt, welche für die Datensatzbearbeitung von Nöten sind.

Der Funktionswert ist vom Typ *integer*.

IDOK	Der Assistent wurde mit "Fertigstellen" beendet.
IDCANCEL	Der Assistent wurde abgebrochen.

Form_RunFileMacro

HWND hwndMain
HWND hwndForm
char * MakroDatei

Diese Funktion startet ein Formularmakro. Das Makro befindet sich in der Datei, deren Name als Parameter *MakroDatei* angegeben wurde.

Als Parameter *hwndMain* muss der Fenster-Handle des Hauptfensters der Anwendung angegeben werden.

Als Parameter *hwndForm* wird der Fenster-Handle des augenblicklich aktiven Formularfensters angegeben. Wenn noch kein Formularfenster eröffnet ist (z.B. weil dieses erst durch den Makrobefehl Öffnen im Makro bewerkstelligt wird), kann hier auch der Wert 0 angegeben werden.

Die Makrobefehle *Öffnen, NeueVorlage* und *Schließen* werden nicht vom Makrointerpreter selbst ausgeführt, sondern müssen von der Rahmenapplikation abgehandelt werden. Wenn einer dieser Befehle vom Makrointerpreter gefunden wird, wird eine Message an das Hauptfenster gesandt, damit die Rahmenapplikation die entsprechende Abhandlung vornehmen kann.

Die Message ist wie folgt parametrisiert:

wParam	lParam	zurückgelieferter Wert
FORMMSG_PERFORMOPEN	DateiName	Handle des neu eröffneten Formular-fensters
FORMMSG_PERFORMNEW	0	Handle des neu eröffneten Formular-fensters
FORMMSG_PERFORMCLOSE	0	Handle des nach dem Schließen aktiven Formularfensters oder 0, wenn danach kein weiteres Formularfenster mehr ak-tiv ist

Der Funktionswert ist vom Typ *integer*.

0 Das Makro wurde fehlerfrei ausgeführt.
-1 Es trat ein Fehler auf. Eine Meldung wurde bereits angezeigt.

Verwandte Funktionen
Form_RunMemMacro

Form_RunMemMacro

HWND hwndMain
HWND hwndForm
char * Makro

Diese Funktion startet ein Formularmakro. Das Makro wird als Textpuffer auf dem Parameter Makro übergeben. Makros, die im Speicher übergeben werden, haben keinen Makrokopf . Die Titelzeile des Makros:

Makro "Titel"

sowie die darauffolgenden Klammern um den Makrorumpf *begin* und *end*, bzw. bei deutscher Schreibweise *anfang* und *ende* entfallen bei Makros, die im Speicher übergeben werden.

Als Parameter *hwndMain* muss der Fenster-Handle des Hauptfensters der Anwendung ange-geben werden.

Als Parameter *hwndForm* wird der Fenster-Handle des augenblicklich aktiven Formularfensters angegeben. Wenn noch kein Formularfenster eröffnet ist (z.B. weil dieses erst durch den Makrobefehl Öffnen im Makro bewerkstelligt wird), kann hier auch der Wert 0 angegeben werden.

Die Makrobefehle *Öffnen, NeueVorlage* und *Schließen* werden nicht vom Makrointerpreter selbst ausgeführt, sondern müssen von der Rahmenapplikation abgehandelt werden. Wenn einer dieser Befehle vom Makrointerpreter gefunden wird, wird eine Message an das Hauptfenster gesandt, damit die Rahmenapplikation die entsprechende Abhandlung vornehmen kann.

Die Parametrisierung der Message ist bei der Funktion *Form_RunFileMacro* beschrieben.

Der Funktionswert ist vom Typ *integer*.

0 Das Makro wurde fehlerfrei ausgeführt.
-1 Es tret ein Fehler auf. Eine Meldung wurde bereits angezeigt.

Verwandte Funktionen
Form_RunFileMacro

Form_Save

HWND hwndForm

Diese Funktion schreibt ein Formular in die Datei zurück, aus der es eröffnet wurde. Diese Datei wird im Titel des Formularfensters angezeigt und heißt Titeldatei. Der Name der Titeldatei kann mit Hilfe der Funktion *Form_GetTitleFileName* abgefragt werden.

Diese Funktion darf nur dann aufgerufen werden, wenn das Formular benannt ist – wenn es also über eine Titeldatei verfügt. Wenn das Formular noch unbenannt ist, z.B. weil es neu angelegt wurde, muss die Funktion *Form-SaveAs* aufgerufen werden.

Nach dem Speichern wird die interne Merkvariable, die anzeigt, ob das Formular verändert wurde (Change-Flag), zurückgesetzt. Diese Variable kann mittels der Funktion *Form_SetChangeFlag* auch explizit gesetzt oder rückgesetzt werden.

Der Funktionswert ist vom Typ *integer*.

0 Das Formular wurde fehlerfrei gespeichert.

-1 Es trat ein Fehler auf. Eine Meldung wurde bereits angezeigt. Das Formular wurde nicht oder nicht vollständig gespeichert.

Verwandte Funktionen

Form_SaveAs
Form_GetTitleFileName
Form_SetChangeFlag

Form_SaveAs_Extended

```
HWND   hwndForm
char *  DateiName
int     reserved
```

Diese Routine speichert ein benanntes oder ein unbenanntes Formular in die Datei, deren Name auf Parameter *DateiName* angegeben wird. Nach dem Speichern wird die angegebene Datei zur Titeldatei. Der Name der Titeldatei kann mit der Funktion *Form_GetTitleFileName* jederzeit abgefragt werden.

Nach dem Speichern wird die interne Merkvariable, die anzeigt, ob das Formular verändert wurde (Change-Flag) zurückgesetzt. Diese Variable kann mittels der Funktion *Form_SetChangeFlag* auch explizit gesetzt oder rückgesetzt werden.

Der Funktionswert ist vom Typ *integer*.

0 Das Formular wurde fehlerfrei gespeichert.

-1 Es trat ein Fehler auf. Eine Meldung wurde bereits angezeigt. Das Formular wurde nicht oder nicht vollständig gespeichert.

Verwandte Funktionen

Form_Save
Form_GetTitleFileName
Form_SetChangeFlag
Form_Export

Form_SaveToProfileExtended

```
HWND hwndForm
char *  fileName
char *  section
long    objTypes
int     flags
```

Diese Funktion schreibt den Inhalt aller benannten Objekte eines Formulars in eine Profile-Datei. Der Name der Profile-Datei wird auf dem Parameter *fileName*, der Name der Sektion, in die geschrieben werden soll, wird auf Parameter s*ection* übergeben.

Der Parameter o*bjTypes* bestimmt, welche Arten von Objekten geschrieben werden sollen. Es können ein oder mehrere Objekttypen durch ein logisches oder miteinander kombiniert werden. Außerdem können folgende Objektgruppen angesprochen werden:

FOTYPE_ALL	es werden alle Objekte behandelt
FOTYPE_INPUT	es werden nur Eingabeobjekte behandelt

Die Liste aller Objekttypen ist im Anhang OBJEKTTYPEN beschrieben.

Alle benannten Objekte mit dem oben angegebenenTyp werden in der Form:

```
ObjektName = ObjektInhalt
```

in die Profile-Datei geschrieben. Objekte ohne Inhalt werden in der Form:

```
ObjektName =
```

in die Profile-Datei geschrieben.

Der Parameter *Mode* bestimmt, wie formatierbare Werte (Ganzzahl, Gleitkommazahl, Datum,...) in die Datei geschrieben werden sollen.
Mögliche Angaben sind:

FORM_PROFILE_NEUTRAL_DATA	Der Inhalt der Objekte wird in neutraler Form notiert: Gleitkommazahlen: Punkt als Dezimaltrenner Datum: YYYY-MM-DD
FORM_PROFILE_FORMATTED_DATA	Der Inhalt der Objekte wird formatiert in die Datei geschrieben. Grundlage für die Formatierung ist die aktuelle Formatangabe des Objektes.

Der Funktionswert ist vom Typ *integer.*

0 Die Funktion wurde fehlerfrei ausgeführt.
-1 Es trat ein Fehler auf. Eine entsprechende Meldung wurde bereits angezeigt.

Verwandte Funktionen
Form_LoadFromProfileExtended

Form_Select

HWND hwndForm
int flags
long objTypes

Mit dieser Funktion können Objekte bestimmter Typen auf der aktuellen Seite markiert werden.

flags ist eine Liste von Bits folgender Optionen:

FM_INPUT	Operation nur für Eingabeobjekte ausführen
FM_GROUPED	Operation nur für gruppierte Objekte ausführen
FM_BEEP	Im Fehlerfall einen Klang ausgeben

FM_UNDO	Ein UnDo-Item absetzen
FM_EDITMODE	Funktion ablehnen, falls nicht der Bearbeitungsmodus eingeschaltet ist
FM_REFRESH	Änderungen sofort anzeigen
FM_NOTIFY	Benachrichtigung über die Aktion an das Hostfenster senden
FM_SHOW_DIALOG	Dialog zur Auswahl der zu selektierenden Objekttypen anzeigen

Die Funktion liefert keinen Funktionswert zurück.

Form_SelectRecordModal

FORM_SELECT_RECORD * *SelectRecord*

Diese Funktion eröffnet einen modalen Dialog, zur Auswahl und Übernahme eines Datensatzes. Die Datenquelle sowie das Datenbankobjekt oder aber auch ein Datenzugriffsformular werden über die Struktur *SelectRecord* an die Funktion übergeben.

Wird der Dialog mit *Übernehmen* bestätigt, und ist in *SelectRecord* ein Dateiname zur Aufnahme des Ergebnisses angegeben, so wird die Information über den ausgewählten.

Die Felder des ausgewählten Datensatzes werden auf mehrere Arten mittels der Funktion *Form_SaveToProfileExtended* in die Zieldatei geschrieben.

Der Funktionswert ist vom Typ *integer*.

IDOK Der Dialog wurde mit *Übernehmen* bestätigt
IDCANCEL Der Dialog wurde abgebrochen.

Form_SetChangeFlag

```
HWND    hwndForm
int     Flag
```

Diese Funktion setzt die interne Merkvariable, ob ein Formular seit dem Öffnen oder Anlegen oder nach dem Speichern verändert worden ist. Wenn die Variable auf verändert steht, wird vor dem Schließen eines Formulars gefragt, ob die Änderungen gespeichert werden sollen oder nicht.

Der Parameter Flag definiert, wie die Variable besetzt werden soll:

0 Formular wurde nicht verändert (frisch eröffnet oder gespeichert).
1 Formular wurde verändert.

Die Funktion liefert keinen Funktionswert zurück.

Verwandte Funktionen
Form_GetChangeFlag

Form_SetColorMode

```
HWND    hwndForm
int     Modus
```

Diese Funktion schaltet ein Formular von der Normaldarstellung in den Schwarz/Weiß-Modus oder umgekehrt. Die Funktion kann sowohl im Ediermodus als auch im Ausfüllmodus jederzeit aufgerufen werden.

Wenn sich ein Formular im Schwarz/Weiß-Modus befindet, werden die Hintergründe aller Objekte weiß und die Ränder und Schriften aller Objekte schwarz dargestellt.

Der Parameter Modus bezeichnet die gewünschte Darstellung:

COLORMODE_NORMAL	Normaldarstellung
COLORMODE_BW	Schwarz/Weiß-Darstellung

Die Funktion liefert keinen Funktionswert zurück.

Verwandte Funktionen
Form_SetColorMode

Form_SetColorProperty

HWND	hwndForm
long	Property
COLORREF	Value

Diese Funktion setzt einen der Farbwerte des durch *hwndForm* spezifizierten Formulars.

Property definiert, welche Farbe auf den Wert *Value* gesetzt werden soll.
Folgende Werte sind möglich:

FORM_PROP_COLOR_WINDOW_BGR	Farbe des Fensters der Anwendung
FORM_PROP_COLOR_PAGE_SHADOW	Farbe des Seiten-Schattens
FORM_PROP_COLOR_ACTIVE_INPUT	Farbe, in der das aktive Eingabefeld dargestellt wird. Die Farbe wird nur verwendet, wenn die Eigenschaft FORM_PROP_ACTIVE_INPUT_DISPLAY mittels der Routine *Form_SetLongProperty* auf den Wert 1 gesetzt wird.
FORM_PROP_COLOR_COMMENT_AVAIL	Farbe, in der Eingabeobjekte dargestellt werden, für die ein Kommentar verfügbar ist
FORM_PROP_COLOR_RASTER	Farbe, in der das Gitternetz angezeigt wird, wenn es eingeschaltet ist

Alle oben aufgeführten Eigenschaften können auch mittels der Funktion *Form_GetColorProperty* ausgelesen werden.

Die Funnktion hat keinen Funktionswert.

Verwandte Funktionen

Form_SetXmlPropertyByAttrName
Form_SetLongProperty
Form_GetColorProperty

Form_SetCurrentInputObject

```
HWND hwndForm
HFOB   hFob
```

Diese Funktion macht das Eingabeobjekt mit dem Handle hFob zum aktiven Eingabeobjekt. Der Cursor wird in das Objekt positioniert. Wenn hFob nicht der Handle eines Eingabeobjektes ist, bleibt der Aufruf der Funktion ohne Wirkung.

Wenn sich das Formular im Ediermodus befindet, ist der Aufruf der Funktion ebenfalls ohne Wirkung.

Die Funktion liefert keinen Funktionswert zurück.

Form_SetCurrentPageBgrMode

```
HWND hwndForm
int      Mode
```

Diese Funktion setzt den Hintergrundmodus der aktuellen Seite des durch hwndForm spezifizierten Formulars.

Mode gibt den einzustellenden Seitenhintergrund an:

PAGEBGR_NONE	Die Seite hat keinen Hintergrund.
PAGEBGR_SOLID	Die Seite hat eine feste Hintergrundfarbe.
PAGEBGR_BITMAP	Die Seite hat ein eingeblendetes Bild als Hintergrund.

Die Funktion liefert keinen Funktionswert zurück.

Form_SetDefaults

```
HWND    hwndForm
int     flags
```

Diese Funktion bewirkt, dass alle Eingabeobjekte eines Formulars mit ihren Vorbesetzungswerten gefüllt werden. Jedes Eingabeobjekt kann beim Anlegen eines Formulars mit einer Vorbesetzung versehen werden. Eingabeobjekte, welche nicht über eine Vorbesetzung verfügen, werden geleert.

Flags ist eine Liste von Bits, mit der sich eine oder mehrere der folgenden Optionen einschalten lässt:

FM_CONFIRM	Die Operation muss vom Anwender quittiert werden
FM_NOREADONLY	Es werden nur Eingabeobjekte vorbesetzt, die nicht *readonly* sind

Der Funktionswert ist vom Typ *integer.*

0 Die Funktion wurde fehlerfrei ausgeführt.
-1 Es trat ein Fehler auf. Eine entsprechende Meldung wurde bereits angezeigt.

Form_SetEditOrderMode

```
HWND    hwndForm
int     Modus
```

Diese Funktion schaltet den Arbeitsmodus ein oder aus, in dem die Reihenfolge der Eingabeobjekte definiert werden kann. Der Parameter Modus hat folgende Bedeutung:

n Der Modus Eingabereihenfolge festlegen wird eingeschaltet. Das erste Objekt, dessen Reihenfolge bestimmt werden soll, ist das Objekt mit der Nummer n. n zählt ab eins.

0 Der Modus Eingabereihenfolge festlegen wird ausgeschaltet. Alle Objekte, die angeordnet wurden, bleiben in der durch den Benutzer bestimmten Reihenfolge.

-1 Der Modus Eingabereihenfolge festlegen wird eingeschaltet. Die Nummer des ersten Objekts, dessen Reihenfolge bestimmt werden soll, wird durch einen Dialog beim Benutzer angefragt.

Die Funktion liefert keinen Funktionswert zurück.

Form_SetGlobalPrintOffsetX

long *Offset*

Die Funktion *Form_SetGlobalPrintOffsetX* setzt den horizontalen globalen Druck-Offset. Dieser Wert kann außer mit dieser Funktion sowohl im Formulareditor als auch im Ausfüllprogramm vom Benutzer eingestellt werden. Er wird auf die horizontalen Positionen aller zu druckenden Objekte aufaddiert, bevor diese gedruckt werden.

Der Offset kann sowohl positiv als auch negativ sein. Ein positiver Offset bewirkt eine Verschiebung aller Objekte nach rechts, ein negativer Offset bewirkt eine Verschiebung aller Objekte nach links.

Der Offset wird in Mikrometern angegeben. Die Angabe 10.000 bewirkt also eine Verschiebung um 1cm nach rechts.

Der Offset gilt für alle Formulare und für alle Drucker. Er ist global. Er bleibt solange bestehen, bis entweder ein neuer Offset eingestellt wird oder bis die DLL neu geladen wird.

Mit der Funktion *Form_EditGlobalPrintOffsets* können sowohl der vertikale als auch der horizontale Druckoffset durch den Benutzer via Dialog eingegeben werden.

Die Funktion liefert keinen Funktionswert zurück.

Verwandte Funktionen

Form_SetGlobalPrintOffsetY
Form_EditGlobalPrintOffsets

Form_SetGlobalPrintOffsetY

long *Offset*

Die Funktion *Form_SetGlobalPrintOffsetY* setzt den vertikalen globalen Druck-Offset. Dieser Wert kann außer mit dieser Funktion sowohl im Formulareditor als auch im Ausfüllprogramm vom Benutzer eingestellt werden. Er wird auf die vertikalen Positionen aller zu druckenden Objekte aufaddiert, bevor diese gedruckt werden.

Der Offset kann sowohl positiv als auch negativ sein. Ein positiver Offset bewirkt eine Verschiebung aller Objekte nach unten, ein negativer Offset bewirkt eine Verschiebung aller Objekte nach oben.

Der Offset wird in Mikrometern angegeben. Die Angabe 10.000 bewirkt also eine Verschiebung um 1cm nach unten.

Der Offset gilt für alle Formulare und für alle Drucker. Er ist also global. Er bleibt solange bestehen, bis entweder ein neuer Offset eingestellt wird oder bis die DLL neu geladen wird.

Mit der Funktion *Form_EditGlobalPrintOffsets* können sowohl der vertikale als auch der horizontale Druckoffset durch den Benutzer via Dialog eingegeben werden.

Die Funktion liefert keinen Funktionswert zurück.

Verwandte Funktionen

Form_SetGlobalPrintOffsetX
Form_EditGlobalPrintOffsets

Form_SetHwndKeyForwarding

HWND hwndForm
HWND hwndKeyForwarding

Mit dieser Routine kann ein Fenster (hwndKeyForwarding) bei dem durch hwndForm bestimmten Formular angemeldet werden, zu welchem alle Tastaturereignisse weitergeleitet werden, die nicht von der Formular-DLL selbst interpretiert werden.

Üblicherweise werden alle Tastaturereignisse an das beim Aufruf von *Form_Open_Expanded* mitgegebene Host-Fenster weitergeleitet. Mit dieser Funktion kann nun ein anderes Fenster bestimmt werden.

Der Funktionswert ist vom Typ HWND.

Der Funktionswert gibt an, zu welchem Fenster-Handle die Tastaturereignisse zuvor weitergeleitet wurden.

Return 0 Es war kein Fenster für Tastaturereignisse bekannt.
Return h Handle des Fensters, an welches die Ereignisse zuvor weitergeleitet wurden

Verwandte Funktionen
Form_SetHwndNotify

Form_SetHwndNotify

HWND hwndForm
HWND hwndNotify

Mit dieser Routine kann ein Fenster (hwndNotify) bei dem durch hwndForm bestimmten Formular angemeldet werden, zu welchem alle Benachrichtigungen seitens der Formular-DLL gesendet werden sollen.

Üblicherweise werden alle Nachrichten an das beim Aufruf von *Form_Open_Expanded* mitgegebene Host-Fenster gesendet. Mit dieser Funktion kann nun ein anderes Fenster bestimmt werden.

Die Funktion liefert keinen Funktionswert zurück.

Verwandte Funktionen
Form_SetHwndKeyForwarding

Form_SetLongProperty

HWND	hwndForm
long	Property
long	Value

Diese Funktion setzt die numerische Eigenschaft *Property* des durch *hwndForm* spezifizierten Formulars auf den Wert *Value*.

Folgende Eigenschaften können gesetzt werden:

FORM_PROP_DISPDIST_HORI	Seitlicher Rand zwischen Fenster und Formularseite in Pixeln
FORM_PROP_DISPDIST_TOP	Oberer Rand zwischen Fenster und Formularseite in Pixeln
FORM_PROP_DISPDIST_BOTTOM	Unterer Rand zwischen Fenster und Formularseite in Pixeln
FORM_PROP_DISPDIST_INTERPAGE	Abstand zwischen zwei Formularseiten in der Anzeige in Pixeln
FORM_PROP_AUTOZOOM_MODE	Autozoom-Modus FORM_AUTOZOOM_NONE kein automatisches Zoomen FORM_AUTOZOOM_WIDTH Zoomen auf Fensterbreite FORM_AUTOZOOM_HEIGHT Zoomen auf Fensterhöhe
FORM_PROP_PAGEKEY_MODE	Blättern-Modus Bestimmt das Verhalten der Tasten *Bild auf* und *Bild ab:* FORM_PAGEKEY_PAGING Blättern auf der Formularseite FORM_PAGEKEY_DBACCESS Blättern durch Datensätze beim Datenbankzugriff

FORM_PROP_HELP_MODE	Hilfe-Modus FORM_HELPMODE_DISABLED Die Hilfefunktionalität ist ausgeschaltet. FORM_HELPMODE_STANDARD Die Standard-Hilfe der Windows-Umgebung wird verwendet. Diese Option ist für Fremdapplikationen gedacht, welche eine eigene Hilfe für das Gesamtprodukt implementieren.
FORM_PROP_DISABLED_COMPONENTS	Bitliste, die einzelne Bedienungs-komponenten der Software ausschaltet. FORMCOMP_BUTTON_MACRO Abschalten der Makroeingabe FORMCOMP_BUTTON_HTMLJAVA Abschalten der Eingabe von Javaskript für HTML-Export FORMCOMP_BUTTON_PDFJAVA Abschalten der Eingabe von Javaskript für PDF-Export FORMCOMP_DATA_ACCESS Abschalten der Datenzugriffsobjekte
FORM_PROP_READONLY_MODE	Formular arbeitet im *Read Only* Modus
FORM_PROP_DISABLE_WARNING_BOXES	Abschalten von Warnungen
FORM_PROP_ACTIVE_INPUT_DISPLAY	1 = aktives Eingabefeld hervorheben 0 = aktives Eingabefeld nicht hervorheben
FORM_PROP_MODAL_HEIGHT	Höhe in Pixeln bei Verwendung des Formulars als modaler Dialog
FORM_PROP_MODAL_WIDTH	Breite in Pixeln bei Verwendung des Formulars als modaler Dialog

FORM_PROP_MODAL_FLAGS	Bitliste mit Optionen für den Betrieb als modaler Dialog: FORM_MODALFLAG_MAXIMIZE Dialog maximieren
FORM_PROP_COLORMODE	0 Darstellung in Farbe COLORMODE_BW Darstellung Schwarz-Weiss
FORM_PROP_GRID	Rasterabstand in Mikrometern
FORM_PROP_RASTER_DISTANCE	Gitternetz-Abstand oder 0 (kein Gitternetz)
FORM_PROP_RASTER_ZORDER	ZORDER_BACKGROUND Gitternetz im Hintergrund anzeigen ZORDER_FOREGROUND Gitternetz im Vordergrund anzeigen
FORM_PROP_NO_SAVE_QUESTION	1 = Frage *Speichern* unterdrücken 0 = Frage *Speichern* nicht unterdrücken
FORM_PROP_NEXTGROUPID	Nächste zu vergebende ID zur Kennzeichnung einer Gruppe von Objekten
FORM_PRINTPROP_COPYLISTMODE	Gibt die Eigenschaft ‚Ausfertigungen drucken' zurück. Ist der Wert != 0, werden alle konfigurierten Ausfertigungen gedruckt, anderenfalls nicht.
FORM_PRINTPROP_ONLYCONTENT	Gibt die Eigenschaft ‚Vordruck ausblenden' zurück. Ist der Wert != 0, wird nur der vom Benutzer eingegebene Inhalt des Formulars gedruckt.
FORM_PRINTPROP_BLACKWHITE	Gibt die Eigenschaft ‚Druck in Schwarz/Weiß' zurück. Ist der Wert != 0, wird das Formular monochrom ausgedruckt.
FORM_PRINTPROP_ZOOM	Gibt die Eigenschaft ‚Vergrößerung' in Prozent zurück.

FORM_PRINTPROP_PAGEMODE	Gibt die Eigenschaft der zu druckenden Seiten zurück. Mögliche Angaben sind: PAGEMODE_ALL: es werden alle Seiten gedruckt PAGEMODE_RANGE: es wird ein Seitenbereich grdruckt PAGEMODE_CURRENT: es wird nur die aktuelle Seite gedruckt
FORM_PRINTPROP_FROMPAGE	Gibt den Wert der ersten zu druckenden Seite zurück, wenn die Eigenschaft FORM_PRINTPROP_PAGEMODE auf den Wert PAGEMODE_RANGE gesetzt ist.
FORM_PRINTPROP_TOPAGE	Gibt den Wert der letzten zu druckenden Seite zurück, wenn die Eigenschaft FORM_PRINTPROP_PAGEMODE auf den Wert PAGEMODE_RANGE gesetzt ist.
FORM_PRINTPROP_COPIES	Gibt die Eingenschaft der ‚Anzahl der zu druckenden Kopieen' zurück.
FORM_PRINTPROP_DUPLEX	Gibt die Eigenschaft des ‚Doppelseitigen Drucks' zurück.
FORM_PRINTPROP_XOFFSET FORM_PRINTPROP_YOFFSET	Gibt die Eigenschaften des horizontalen bzw. vertikalen Druckoffsets zurück. Diese sind nicht zu verwechseln mit den global (im Editor oder Filler) eingestellten, über alle Formulare hinweg gültigen Offsets. Die hier eingestellten Werte werden formularspezifisch zu den globalen Werten hinzu addiert.

Der Funktionswert ist vom Typ *integer.*

IDOK Die Eigenschaft wurde wie gewünscht gesetzt.
IDCANCEL Die Eigenschaft konnte nicht gesetzt werden.

Verwandte Funktionen

Form_GetLongProperty
Form_SetStringProperty

Form_SetColorProperty
Form_SetXmlPropertyByAttrName

Form_SetObjectNamesPool

HWND hwndForm
char * Liste

Diese Funktion definiert eine Liste von Objektnamen, welche in den Dialogen zum Ändern und Anlegen von Objekten als Combobox zur Namenseingabe für das Objekt angeboten wird.

Der Parameter Liste ist ein String, der alle Objekte namentlich, durch ein Komma getrennt, aufzählt.

Die Funktion liefert keinen Funktionswert zurück.

Form_SetPrinterBin

HWND hwndForm
int Page
int Bin

Diese Funktion setzt den Index des zu verwendenden Druckerschachts, der beim Ausdruck einer Seite des Formulars verwendet werden soll.

Der Parameter *Page* bestimmt die Seite, auf die sich die Schachtangabe bezieht. Die Numerierung der Seiten beginnt mit 1. Wenn der Wert –1 angegeben wird, werden die Schächte aller Seiten verändert.

Der Parameter *Bin* bestimmt den Index des Schachtes, der gesetzt werden soll. Der Index bezieht sich auf die druckerspezifisch vom Drucker gelisteten Schächte. Der Index 0 entspricht dem ersten Schacht. Wenn der Wert –1 angegeben wird, bedeutet dies, dass das Papier aus dem Standardschacht eingezogen werden soll.

Die Funktion liefert keinen Funktionswert zurück.

Verwandte Funktionen
Form_SetPrinterNameAndPort
Form_SetPrinterMode

Form_SetPrinterMode

```
HWND    hwndForm
int     Mode
```

Diese Funktion setzt die interne Variable des Formulars, welche darüber entscheidet, ob das Formular beim Druck auf den Standarddrucker oder auf den im Drucker eingestellten Druckernamen ausgegeben wird.

Der Parameter Mode kann folgende Werte annehmen:

```
PRINTERMODE_EXPLICIT
PRINTERMODE_STANDARD
```

Die Funktion liefert keinen Funktionswert zurück.

Verwandte Funktionen

Form_SetPrinterNameAndPort
Form_SetPrinterBin

Form_SetPrinterNameAndPort

```
HWND          hwndForm
const char *  Name
const char *  Port
```

Diese Funktion setzt einen neuen Druckernamen und einen neuen Druckerport für das Formular. Wenn das Formular gedruckt wird, werden diese Angaben verwendet.

Der Parameter *Name* ist ein String, der den Druckernamen definiert.
Der Parameter *Port* ist ein String, der den Port für die Ausgabe definiert.

Die Funktion liefert keinen Funktionswert zurück.

Verwandte Funktionen

Form_SetPrinterBin
Form_SetPrinterMode

Form_SetPrintExpiration

```
HWND    hwndForm
long    Expiration
```

Diese Funktion setzt das Verfallsdatum für das durch hwndForm spezifizierte Formular. Nach Ablauf des Datums kann das Formular nicht mehr ausgedruckt werden. Expiration gibt das Verfallsdatum in einer Anzahl an Tagen seit dem 1.1.1600 an.

Die Funktion liefert keinen Funktionswert zurück.

Form_SetWorkModeExt

```
HWND    hwndForm
int     mode
int     flags
```

Diese Funktion ändert den Arbeitsmodus eines Formulars. Wenn sich das Formular im Ausfüllmodus befindet, kann es in den Ediermodus umgeschaltet werden – und umgekehrt.

Der Parameter *mode* bestimmt den neuen Arbeitsmodus:

```
FORMMODE_EDIT   Es wird in den Ediermodus umgeschaltet.
FORMMODE_RUN    Es wird in den Ausfüllmodus umgeschaltet.
FORMMODE_VIEW   Es wird in den Read-Only-Ausfüllmodus umgeschaltet.
```

Beim Umschalten eines Formulars vom Ausfüllmodus in den Ediermodus geht die Information, welches das aktuelle Eingabeobjekt ist, verloren, da es im Ediermodus keine aktiven Eingabeobjekte gibt.

Flags ist eine Bitliste folgender Werte:

FM_FIRST	Beim Umschalten in Ausfüllmodus wird das erste Eingabeobjekt des Formulars wird zum aktiven Eingabeobjekt.
FM_VISIBLE	Beim Umschalten in Ausfüllmodus wird das erste Eingabeobjekt im sichtbaren Bildschirmausschnitt zum aktiven Eingabeobjekt.

Der Funktionswert ist vom Typ *integer,*

0 Die Funktion wurde korrekt durchgeführt.
-1 Es ist ein Fehler aufgetreten. Eine Fehlermeldung wurde bereits angezeigt.

Verwandte Funktionen
Form_GetWorkMode

Diese Funktion ersetzt folgende ältere Funktionen
Form_SetWorkMode

Die älteren Funktionen sollten nicht mehr verwendet werden.

Form_SetXmlPrintProperty

HWND	hwndForm
const char *	attr
char *	data

Diese Funktion setzt eine Druck-Eigenschaft des Formulars *hwndForm.* Die Eigenschaft wird durch den Attribut-Namen *attr* spezifiziert. Der Wert der Eigenschaft wird unabhängig vom Typ der Eigenschaft immer als Zeichenkette auf dem Parameter *data* übergeben.

Alle Druck-Eigenschaften sind im Anhang PRINT XML PROPERTIES beschrieben.

Der Funktionswert ist vom Typ *integer*

0 Die Eigenschaft wurde wie gewünscht gesetzt.
-1 Die Eigenschaft *attr* ist unbekannt und konnte nicht gesetzt werden.

Verwandte Funktionen
Form_SetXmlPropertyByAttrName
Form_GetXmlPrintProperty

Form_SetXmlPropertyByAttrName

HWND	hwndForm
const char *	AttrName
const char *	String

Diese Funktion setzt eine Eigenschaft des Formulars *hwndForm.* Die Eigenschaft wird durch den Attribut-Namen *AttrName* spezifiziert. Der Wert der Eigenschaft wird unabhängig vom Typ der Eigenschaft immer als Zeichenkette auf dem Parameter *String* übergeben.

Wenn die Eigenschaft numerisch (ganzzahlig) ist, wird der numerische Wert als String übergeben, z.B. "123".

Wenn die Eigenschaft eine Farbe ist, wird der Wert als String der Form RRR,GGG,BBB übergeben, wobei die drei Komponenten RRR, GGG und BBB die numerischen Werte der Farben *Rot, Grün* und *Blau* in dezimaler Darstellung repräsentieren. 255,255,000 repräsentiert beispielsweise die Farbe *Gelb.*

Alle Formular-Eigenschaften sind im Anhang FORM XML PROPERTIES beschrieben.

Alle Eigenschaften können mit Hilfe der Funktion *Form_GetXmlPropertyByAttrName* auch gelesen werden.

Der Funktionswert ist vom Typ *integer*

0	Die Eigenschaft wurde wie gewünscht gesetzt.
-1	Die Eigenschaft *AttrName* ist unbekannt oder konnte nicht gesetzt werden.

Verwandte Funktionen

Form_SetXmlProperty
Form_SetLongProperty
Form_SetColorProperty
Form_GetXmlPropertyByAttrName

Form_SetZoomFactor

HWND hwndForm
int factor

Diese Funktion setzt einen neuen Zoom-Faktor.

factor Der Faktor ist entweder ein expliziter Wert in Prozent oder eine der folgenden Metagrößen:

ZOOMMODE_HALFWIDTH	Die Hälfte der aktuellen Seitenbreite füllt das Fenster
ZOOMMODE_WIDTH	Die ganze Seitenbreite füllt das Fenster
ZOOMMODE_DOUBLEWIDTH	Zwei Seiten nebeneinander füllen das Fenster, die Höhe kann angeschnitten sein
ZOOMMODE_ONE_PAGE	Eine ganze Seite passt in voller Höhe ins Fenster
ZOOMMODE_TWO_PAGES	Zwei Seiten werden in voller Höhe nebeneinander dargestellt
ZOOMMODE_THREE_PAGES	Drei Seiten werden in voller Höhe nebeneinander dargestellt.
ZOOMMODE_MULTI_PAGES	Es werden mehrere Seiten dargestellt.
ZOOMMODE_DIALOG	Der Zoomfaktor wird via Dialog angefragt.
ZOOMMODE_ZOOM_IN	Der Zoom wird um eine Stufe vergrößert.
ZOOMMODE_ZOOM_OUT	Der Zoom wird um eine Stufe verkleinert.

Die Funktion liefert als Funktionswert immer 0 zurück.

Diese Funktion ersetzt folgende veraltete Funktionen
Form_EditZoomFactor

Verwandte Funktionen
Form_GetZoomFactor

Form_SplitObjSet

HWND	hwndForm
char *	FileName
long	Mode

Die Funktion erzeugt ein neues einseitiges Formular. Auf der ersten Seite dieses Formulars werden alle im aktuell geöffneten Formular (*hwndForm*) selektierten Objekte eingefügt.

Der Name des zu generierenden Formulars wird auf dem Parameter *FileName* übergeben. *Mode* ist für zukünftige Zwecke gedacht und muss mit 0 vorbesetzt werden.

Der Funktionswert ist vom Typ *integer*.

IDOK	Das Formular wurde erzeugt, die Objekte wurden fehlerfrei übertragen.
IDCANCEL	Es ist ein Fehler aufgetreten.

Verwandte Funktionen

Form_AppendObjSet

Form_ToggleColorMode

HWND	hwndForm
int	Mode

Diese Funktion schaltet einzelne Einstellungen der Farbkonfiguration eines Formulars um. Zur Zeit ist es nur möglich in den Farb- bzw. in den Schwarz/Weiß-Modus zuschalten. Dazu muss Mode auf den Wert COLORMODE_BW gesetzt werden.

Die Funktion liefert keinen Funktionswert zurück..

Form_VerifySemantic

HWND hwndForm
int flags

Diese Funktion prüft die semantische Richtigkeit aller Eingaben in allen Eingabeobjekten. Die Funktion sollte nur im Ausfüllmodus aufgerufen werden.

Flags ist eine Liste von Bits, mit der sich eine oder mehrere der folgenden Optionen einschalten lässt:

FM_BEEP	Im Fehlerfall einen Klang ausgeben
FM_EDITMODE	Funktion ablehnen, falls nicht der Bearbeitungsmodus eingeschaltet ist

Der Funktionswert ist vom Typ *HFOB.*

0 Alle Eingabeobjekte haben einen semantisch korrekten Inhalt.

xx Mindestens ein Eingabeobjekt enthält eine unzulässige Eingabe.
xx ist der Handle des ersten fehlerhaften Eingabeobjektes.

Form_WantHotKey

HWND	hwndForm
int	key
bool	control
bool	shift

Diese Funktion deklariert einem Formular den Wunsch der Host-Anwendung, dass Taste mit dem Wert *key* nicht vom Formular verarbeitet werden sollen, sondern als Notifikation vom Typ *FORMMSG_HOTKEY* an das Host-Fenster gesendet wird.

Wenn für *control* der Wert *true* übergeben wird, gilt dies nur für Tasten, die in Verbindung mit der Taste *Strg* gedrückt werden.

Wenn für *shift* der Wert *true* übergeben wird, gilt dies nur für Tasten, die in Verbindung mit der Taste *Umschalt* gedrückt werden.

Beide Schalter *control* und *shift* können auch gleichzeitig gesetzt werden. Ist keiner der beiden Schalter gesetzt, wird die Notifikation nur gesendet, wenn die Taste tatsächlich ohne Zusatztaste gedrückt wird.

Der Funktionswert ist vom Typ *integer*.

0	Die Taste wurde angemeldet.
-1	Fehler (Falscher Handle oder zu viele Tasten angemeldet).

Bitte lesen Sie die Beschreibung der Notifikation FORMMSG_HOTKEY, um zu erfahren, in welcher Form der Tastenwert und die Zusatztasten übergeben werden.

Form_XmlObjectWizzard

HWND	hwndForm
long	Flags

Diese Funktion startet den Assistenten für neue Objekte basierend auf XML-Vorlagen analog zur gleichnamigen Menüfunktion des Formulareditors. *Flags* ist für zukünftige Zwecke gedacht und muss mit 0 vorbesetzt werden.

Der Funktionswert ist vom Typ *integer*.

IDOK	Der Assistent wurde mit Objekt(e) einfügen bestätigt.
IDCANCEL	Der Assistent wurde abgebrochen.

5. Seitenfunktionen

Dieses Kapitel beschreibt alle Funktionen, die die Kontrolle einer einzelnen Formular-Seite ermöglichen. Diese Funktionen werden *Seitenfunktionen* genannt.

Alle *Seitenfunktionen* werden mit einem *Seiten-Handle* versorgt, welcher die Seite, auf die sich die Funktion bezieht, beschreibt. Ein Seiten-Handle ist eine Variable vom Typ **HFPAGE**.

Der Handle einer Seite kann auf verschiedene Weise ermittelt werden. Beispielsweise liefert die Funktion *Form_GetPageHandle* den Handle zu einer bestimmten Seitennummer zurück. Die Routine *Form_PixelToPos* liefert einen Handle zu einer Seite zurück, welche über eine Bildschirmposition (z.B. den Mauszeiger) bestimmt wurde.

FormPage_GetXmlPropertyByAttrName

HFPAGE	hPage
const char *	AttrName
char *	Result
int	Size

Diese Funktion liefert eine Eigenschaft einer Formularseite zurück. Die Seite wird durch den Seiten-Handle *hPage* spezifiziert. Die Eigenschaft wird durch den Attribut-Namen *AttrName* spezifiziert.

Der Wert der Eigenschaft wird unabhängig vom Typ der Eigenschaft immer als Zeichenkette auf dem Parameter *Result* zurückgeliefert. Es werden nicht mehr als *Size* Zeichen übertragen. Das Ergebnis ist ist immer nullterminiert.

Wenn die Eigenschaft numerisch (ganzzahlig) ist, wird der numerische Wert als String uirückgeliefert, z.B. "123".

Wenn die Eigenschaft eine Farbe ist, wird der Wert als String der Form R,G,B übergeben, wobei die drei Komponenten R, G und B die numerischen Werte der Farben *Rot, Grün* und *Blau* in dezimaler Darstellung repräsentieren. 255,255,000 repräsentiert beispielsweise die Farbe *Gelb.*

Alle Seiten-Eigenschaften sind im Anhang PAGE XML PROPERTIES beschrieben.

Alle können mit Hilfe der Funktion *FormPage_SetXmlPropertyByAttrName* auch gesetzt werden.

Der Funktionswert ist vom Typ *integer*

0 Die Eigenschaft wurde wie gewünscht zurückgeliefert.
-1 Die Eigenschaft *AttrName* ist unbekannt oder konnte nicht gesetzt werden.

Verwandte Funktionen
FormPage_SetXmlPropertyByAttrName
FormObj_GetXmlPropertyByAttrName
Form_GetXmlPropertyByAttrName

FormPage_SetXmlPropertyByAttrName

HFPAGE	hPage
const char *	AttrName
const char *	String

Diese Funktion setzt eine Eigenschaft einer Formularseites. Die Seite wird durch den Seiten-Handle *hPage* spezifiziert. Die Eigenschaft wird durch den Attribut-Namen *AttrName* spezifiziert. Der Wert der Eigenschaft wird unabhängig vom Typ der Eigenschaft immer als Zeichenkette auf dem Parameter *String* übergeben.

Wenn die Eigenschaft numerisch (ganzzahlig) ist, wird der numerische Wert als String übergeben, z.B. "123".

Wenn die Eigenschaft eine Farbe ist, wird der Wert als String der Form RRR,GGG,BBB übergeben, wobei die drei Komponenten RRR, GGG und BBB die numerischen Werte der Farben *Rot, Grün* und *Blau* in dezimaler Darstellung repräsentieren. 255,255,000 repräsentiert beispielsweise die Farbe *Gelb.*

Alle Formular-Eigenschaften sind im Anhang FORM XML PROPERTIES beschrieben.

Alle Eigenschaften können mit Hilfe der Funktion *FormPage_GetXmlPropertyByAttrName* auch gelesen werden.

Der Funktionswert ist vom Typ *integer*

0 Die Eigenschaft wurde wie gewünscht gesetzt.
-1 Die Eigenschaft *AttrName* ist unbekannt oder konnte nicht gesetzt werden.

Verwandte Funktionen

FormPage_GetXmlPropertyByAttrName
FormObj_SetXmlPropertyByAttrName
Form_SetXmlPropertyByAttrName

6. Objektfunktionen

Dieses Kapitel beschreibt alle Funktionen, die die Kontrolle eines einzelnen Formularobjekts ermöglichen. Diese Funktionen werden *Objektfunktionen* genannt.

Alle *Objektfunktionen* werden mit einem *Objekt-Handle* versorgt, welcher das Objekt, auf das sich die Funktion bezieht, beschreibt. Ein Objekt-Handle ist eine Variable vom Typ *HFOB*.

Der Handle eines Objektes lässt sich auf verschiedene Arten ermitteln. Wenn man den Namen des Objekts kennt, ist die Funktion *Form_FindNextObject* sehr hilfreich. Mittels dieser Funktion können auch in einer Schleife alle Objekte nacheinander angesprochen werden.

Die Funktion *Form_GetCurrentInputObject* liefert den Handle des augenblicklich aktiven Eingabefeldes zurück.

FormObj_GetFontFaceName

HFOB	hFob
char *	Result
int	Size

Diese Funktion liefert den Namen der Schriftart des Objekts mit dem Handle *hFob* zurück. Wenn das Objekt über keine Schriftart verfügt, wird ein Leerstring zurückgeliefert.

Das Ergebnis wird in *Result* übergeben. Es werden nicht mehr als *Size* Zeichen übertragen. Das Ergebnis ist in jedem Fall nullterminiert.

Die Funktion liefert keinen Funktionswert zurück.

Verwandte Funktionen

FormObj_GetFontHeight
FormObj_GetFontHandle

FormObj_GetFontHandle

HFOB *hFob*

Diese Funktion liefert den *Font Handle* des Objekts mit dem Handle *hFob* zurück. Mit Hilfe des *Font Handles* können Eigenschaften des Fonts gelesen oder gesetzt werden. Hierzu stehen spezielle *Fontfunktionen* zur Verfügung.

FormObj_GetFontHeight

HFOB *hFob*

Diese Funktion liefert die Höhe der Schriftart des Objekts mit dem Handle *hFob* zurück. Wenn das Objekt über keine Schriftart verfügt, wird der Wert 0 zurückgeliefert.

Der Funktionswert ist vom Typ *integer.*

0 *hFob* ist kein gültiger Objekt-Handle oder das Objekt verfügt über keine Schriftart und Schriftgröße.

n Die Schriftart des Objekts mit dem Handle *hFob* hat eine Höhe von n Zehntelpunkten.

Verwandte Funktionen
FormObj_GetFontFaceName

FormObj_GetIndexInComboList

HFOB hObjekt
char * Text

Diese Funktion prüft, ob *Text* in der Auswahlliste des Auswahlfeldes mit dem Handle *hObjekt* enthalten ist und wenn ja, in welcher Position er in der Liste steht.

Der Funktionswert ist vom Typ *integer.*

n Position, in der der Text in der Liste vorkommt (zählweise ab Null)
-1 Der Text kommt nicht in der Auswahlliste vor

FormObj_GetState

HFOB *hFob*

Diese Funktion liefert die Statusbits des Objekts mit dem Handle *hFob* als *long* zurück.

Verwandte Funktionen
FormObj_SetState

FormObj_GetXmlPropertyByAttrName

HFOB	hFob
const char *	AttrName
char *	Result
int	Size

Diese Funktion liefert eine Eigenschaft eines Formularobjekts zurück. Das Objekt wird durch den Handle *hFob* spezifiziert. Die Eigenschaft wird durch den Attribut-Namen *AttrName* spezifiziert.

Der Wert der Eigenschaft wird unabhängig vom Typ der Eigenschaft immer als Zeichenkette auf dem Parameter *Result* zurückgeliefert. Es werden nicht mehr als *Size* Zeichen übertragen. Das Ergebnis ist immer nullterminiert.

Wenn die Eigenschaft numerisch (ganzzahlig) ist, wird der numerische Wert als String uirückgeliefert, z.B. "123".

Wenn die Eigenschaft eine Farbe ist, wird der Wert als String der Form R,G,B übergeben, wobei die drei Komponenten R, G und B die numerischen Werte der Farben *Rot, Grün* und *Blau* in dezimaler Darstellung repräsentieren. 255,255,000 repräsentiert beispielsweise die Farbe *Gelb.*

Alle Objekt-Eigenschaften sind im Anhang OBJECT XML PROPERTIES beschrieben.

Alle Eigenschaften können mit Hilfe der Funktion *FormObj_SetXmlPropertyByAttrName* auch gesetzt werden.

Der Funktionswert ist vom Typ *integer*

0 Die Eigenschaft wurde wie gewünscht zurückgeliefert.
-1 Die Eigenschaft *AttrName* ist unbekannt oder konnte nicht gesetzt werden.

Verwandte Funktionen
FormObj_SetXmlPropertyByAttrName
FormPage_GetXmlPropertyByAttrName
Form_GetXmlPropertyByAttrName

FormObj_InsertToControl

HFOB	*hFob*
const char *	*text*
long	flags

Diese Funktion fügt in das Windows-Control des augenblicklich aktiven Eingabefeldes eine Textsequenz ein. Wenn sich das Formular im Bearbeitungsmodus befindet oder wenn das Eingabeobjekt nicht aktiv ist, ist die Funktion keine Wirkung.

Der einzufügende Text wird als Parameter *text* übergeben. Wenn in dem Eingabefeld nichts markiert ist, wird der an der aktuellen Cursorposition des Eingabefeldes eingefügt. Wenn in dem Eingabefeld eine Markierung besteht, wird der markierte Text entfernt und durch den übergebenen Text ersetzt.

Der Parameter *flags* ist für spätere Zwecke reserviert und muss 0 sein.

Der Funktionswert ist vom Typ *integer*

0 Es wird immer der Wert 0 zurückgeliefert.

FormObj_SetState

HFOB	hFob
long	State

Diese Funktion setzt alle Statusbits des Objekts mit dem Handle *hFob* als *long value.*

Die Funktion liefert keinen Funktionswert zurück.

Verwandte Funktionen
FormObj_GetState

FormObj_SetText

HFOB	hObjekt
char *	Text
int	Notification
int	Makroaufruf

Diese Funktion setzt einen neuen Text in ein beliebiges Objekt mit dem Handle *hObjekt*. Der Text wird als Parameter *Text* übergeben. Wenn der Parameter den Wert NULL hat, wird der Text des Objektes gelöscht.

Wenn ein Text in Eingabe- oder Auswahlfelder geschrieben wird, die über eine Formatierung verfügen, wie z.B. Datum, Uhrzeit, Gleit- oder Festkommazahl, ist darauf zu achten, dass der Text der Syntax und Semantik des Feldes entspricht.

Der Text wird beim Eintragen automatisch gemäß dem eingestellten Format formatiert.

Ankreuzfelder werden angekreuzt, wenn Text dem im Ankreuzfeld definierten Eintrag wenn angekreuzt entspricht. Entsprechend wird das Kreuz ausgeschaltet, wenn Text dem Eintrag wenn nicht angekreuzt entspricht.

Der Parameter *Notification* bestimmt, ob nach dem Setzen des neuen Textes in das Objekt eine Notification-Message an das Hauptfenster gesendet werden soll, so wie dies geschieht, wenn beim Ausfüllen der Inhalt eines Objekts verändert wird. Es können folgende Werte angegeben werden:

NOTIFY_ON	Notification senden
NOTIFY_OFF	keine Notification senden

Der Parameter *Makroaufruf* bestimmt, ob nach dem Setzen des neuen Textes in das Objekt das Änderungsmakro aufgerufen werden soll, sofern es sich um ein Eingabeobjekt handelt.

MACROCALL_ON	Änderungsmakro aufrufen
MACROCALL_OFF	Änderungsmakro nicht aufrufen

Die Funktion liefert keinen Funktionswert zurück.

FormObj_SetXmlPropertyByAttrName

HFOB	hFob
const char *	AttrName
const char *	String

Diese Funktion setzt eine Eigenschaft eines Formularobjekts. Das Objekt wird durch den Handle *hFob* spezifiziert. Die Eigenschaft wird durch den Attribut-Namen *AttrName* spezifiziert. Der Wert der Eigenschaft wird unabhängig vom Typ der Eigenschaft immer als Zeichenkette auf dem Parameter *String* übergeben.

Wenn die Eigenschaft numerisch (ganzzahlig) ist, wird der numerische Wert als String übergeben, z.B. "123".

Wenn die Eigenschaft eine Farbe ist, wird der Wert als String der Form R,G,B übergeben, wobei die drei Komponenten R, G und B die numerischen Werte der Farben *Rot, Grün* und *Blau* in dezimaler Darstellung repräsentieren. 255,255,000 repräsentiert beispielsweise die Farbe *Gelb.*

Alle Objekt-Eigenschaften sind im Anhang OBJECT XML PROPERTIES beschrieben.

Alle Eigenschaften können mit Hilfe der Funktion *FormObj_SetXmlPropertyByAttrName* auch wieder abgefragt werden.

Der Funktionswert ist vom Typ *integer*

0	Die Eigenschaft wurde wie gewünscht gesetzt.
-1	Die Eigenschaft *AttrName* ist unbekannt oder konnte nicht gesetzt werden.

Verwandte Funktionen

FormObj_GetXmlPropertyByAttrName
FormPage_SetXmlPropertyByAttrName
Form_SetXmlPropertyByAttrName

FormObj_ShowLongComment

HFOB	*hFob*
int	*Mode*

Diese Funktion zeigt den langen Kommentar eines Eingabeobjekts in einem Fenster an. Wenn das Objekt über keinen langen Kommentar verfügt, wird eine Fehlermeldung angezeigt.

Der Parameter *Mode* ist eine Bitliste und kann folgende Werte annehmen:

SLC_MODALDIALOG	Der Kommentar wird als modaler Dialog angezeigt
SLC_NOEMPTYERR	Es wird keine Fehlermeldung ausgegeben, wenn das Objekt über keinen langen Kommentar verfügt

Wenn mehr als ein Wert angegeben wird, sind die Angaben durch ein logisches *oder* miteinander zu verknüpfen.

Die Funktion liefert keinen Funktionswert zurück.

7. Einbindung in ein C++ oder C-Projekt

Dieses Kapitel soll dem Entwickler helfen, die Formular-DLL in eine Rahmenapplikation einzubinden, die in C++ oder C programmiert ist. Als Anschauungsobjekt wird der Quellcode einer Beispielapplikation mit dem Namen *formtest* diskutiert und erläutert. Ziel dieser Erläuterung ist es nicht, jede Quellzeile einzeln zu kommentieren. Vielmehr sollen die spezifischen Aufrufe der Formular-DLL herausgearbeitet werden.

Der Quellcode wird bei der Installation nebst allen benötigten Header- und Bibliotheksdateien in das Unterverzeichnis *cpp* des Verzeichnisses kopiert, das als Ziel der Installation angegeben wurde. Der Quellcode kann kompiliert und ausgeführt werden. Der Quellcode darf auch modifiziert werden und als Basis für eigene Entwicklungen dienen. Er ist frei von Lizenzen und darf uneingeschränkt zur Entwicklung benutzt werden, wenn der Entwicklungskit der Formularsoftware ordnungsgemäß lizenziert wurde.

Bei der Testapplikation handelt es sich um ist eine herkömmliche MDI-Applikation auf Basis des Windows-API. Es werden keine Klassenbibliotheken benutzt, um die Applikation möglichst transparent zu machen.

Compiler-Optionen

Bei den Optionen für den Compiler ist zu berücksichtigen, dass alle Strukturen der Bibliotheken der Formularverwaltung mit dem Alignment *1 Byte* compiliert sind. Dementsprechend sollte die Applikation, in die die Formularverwaltung eingebunden wird, möglichst die gleiche Option verwenden.

7.1. Analyse des Beispiel-Quellcodes

Die Analyse beschränkt sich auf die Quelle ***formtest.cpp*** beleuchtet. Zum besseren Verständnis beim Lesen sind die meisten Unterprogrammnamen nach einem Namensschema konstruiert. Der Anfang des Unterprogrammnamens gibt Aufschluss über die Bedeutung der Funktion:

Form_...	Unterprogramme der Formular-DLL
Handle_MsgMain_...	Abhandlung von Messages an das Hauptfenster
Handle_MsgMdi_...	Abhandlung von Messages an MDI-Fenster
Handle_Ac_Main...	Abhandlung von Kommandos des Hauptfensters

Die Funktion WinMain

Da die Applikation mittels API-Schnittstellen formuliert ist, soll mit der Erläuterung der Funktion *WinMain* begonnen werden, welche in jeder Windows-Applikation als zentraler Einsprungpunkt vorhanden sein muss:

```
int PASCAL WinMain (
        HINSTANCE hInstance,
        HINSTANCE hPrevInstance,
        LPSTR lpszCmdParam,
        int nCmdShow)
```

Gleich zu Anfang der Routine wird die Struktur *Global*, welche die meisten der statischen Variablen der Applikation beinhaltet, vorbesetzt.

Einige Zeilen später wird eine globale Message registriert, die als Kommunikation zwischen der Rahmenapplikation und der Formular-DLL dient. Diese wird auf einer globalen Variablen gespeichert:

```
Global.FormMessage = RegisterWindowMessage (MESSAGE_FORM);
```

Es werden einige globale Parameter aus der Profiledatei gelesen. Ferner werden die Menüs und Accelerator-Tabellen geladen. Schließlich wird das Hauptfenster der Applikation eröffnet. Der Handle des Fensters wird ebenfalls auf einer globalen Variablen gespeichert:

```
Global.hwndMain = CreateWindow ( ...
```

Nach weiteren Initialisierungen wird schließlich die Message-Loop erreicht. Diese lautet:

```
while (GetMessage (&msg, NULL, 0, 0) ....
```

Hier fällt insbesondere die Zeile mit dem Funktionsaufruf:

```
if (!Form_IsCurrentObjectMultilineEdit (hwndForm) ...
```

auf. Diese Zeile prüft für jedes eingegebene Zeichen, ob sich der Cursor in einem mehrzeiligen Eingabefeld befindet. Wenn ja, werden einige Tasten, die normalerweise zur MDI-Steuerung benutzt werden, wie z.B. Ctrl+Tab, nicht verteilt, sondern unverändert gelassen, damit sie ungehindert in das mehrzeilige Eingabefeld gelangen, wo sie eine lokal andere Bedeutung haben. Wenn diese Zeile der Message-Loop weggelassen wird, funktionieren diese Tasten in den

mehrzeiligen Eingabefeldern der Formularverwaltung nicht, sondern sie haben die gewohnte übergeordnete Bedeutung zur Steuerung der MDI-Fenster.

Nach dem Verlassen der Message-Loop wird die Rahmenapplikation unter Abschluss aller offenen Prozesse beendet.

Die Callback-Funktion des Hauptfensters

Die Callback-Funktion des Hauptfensters wird durch folgenden Prozedurkopf eingeleitet:

```
CBFUNC (long) WndProcMain
        (HWND hwnd, MSGPARAM message,
        WPARAM wParam, LPARAM lParam)
```

Das Makro CBFUNC (Typ) ist in der Headerdatei der Formularverwaltung definiert. Es wird verwendet, um die Quellhaltung syntaktisch für die 16- und 32-Bit-Welt kompatibel zu halten.

Gleich zu anfang der Callback-Funktion wird die dynamisch registrierte Message zur Koordination der Rückmeldungen von der Formular-DLL an das Hauptfenster abgehandelt. Es wird die Routine

```
Handle_MsgMain_FormMessage (wParam, lParam);
```

aufgerufen, welche die Messages der Formular-DLL an das Hauptfenster zentral abhandelt. Diese Routine wird später beschrieben.

Ferner fällt die Behandlung der lokal definierten Message *WM_OWNMDIACTIVATE* auf. Die Behandlung erfolgt durch das zentrale Unterprogramm:

```
Handle_MsgMain_OwnMdiActivate ();
```

Diese Message wird immer dann abgesetzt, wenn entweder das Hauptfenster der Applikation eine Message vom Typ *WM_ACTICATE* oder wenn ein MDI-Fenster eine Message vom Typ *WM_MDIACTIVATE* erhält.

Die zugeordnete Routine trägt Sorge dafür, dass das entsprechende Formularfenster korrekt aktiviert wird.

Alle weiteren Messages der Callback-Funktion des Hauptfensters sind im wesentlichen Standard-Messages und sollen hier nicht weiter diskutiert werden, da sie nichts mit der eigentlichen Formularverwaltung zu tun haben.

Die Routine Handle_MsgMain_OwnMdiActivate

Wie oben beschrieben, behandelt diese Routine die Messages vom Typ *WM_OWNMDIACTIVATE* an das Hauptfenster. Diese wird immer dann abgesetzt, wenn entweder das Hauptfenster eine Message vom Typ *WM_ACTICATE* oder wenn ein MDI-Fenster eine Message vom Typ *WM_MDIACTIVATE* erhält.

Die Routine ermittelt zunächst das augenblicklich aktive MDI-Fenster durch den Aufruf:

```
SendWmMdiGetActive (Global.hwndClient, NULL);
```

Dann wird aus dem Long-Pointer des Fensters die Adresse der Kontrollstruktur MdiState, gelesen, welche an alle Formular-MDI-Fenster angeheftet ist. Dieser Kontrollstruktur wird dann der Formularfenster-Handle entnommen. Nach einem Sicherheitstest, ob dieser Handle wirklich ein Formularfenster ist, wird der Formularverwaltung mittels des Aufrufes *Form_ActivateFormWindow* mitgeteilt, dass eben dieses Formular von nun an das aktive sein wird. Das ganze geschieht durch die Quellcodesequenz:

```
if ((MdiState = (MDISTATE *) GetWindowLong
        (Global.hwndActiveMdi, 0)) != NULL)
        if (Form_IsFormWindow (MdiState->hwndForm))
                Form_ActivateFormWindow (MdiState->hwndForm);
```

Sollte *kein* Formularfenster aktiv sein, wird der Menübalken auf *kein MDI* umgestellt:

```
SendWmMdiSetMenu
        (Global.hwndClient,
        Global.hMenuNoMdi,
        (HMENU) 0);
```

Die Routine Handle_MsgMain_Command

Die Funktion *Handle_MsgMain_Command* behandelt, wie der Name sagt, Messages vom Typ *WM_COMMAND* für das Hauptfenster der Applikation. Hier gehen insbesondere alle Menü-Aktionen der Applikation ein.

Die Struktur *WMCOMMAND* und der Aufruf *Evaluate_WmCommand* sind Hilfsmittel, um die Behandlung der Messages vom Typ *WM_COMMAND* quellcodekompatibel für die 16- und 32-Bit-Welt zu kodieren.

Die eingehenden Kommandos werden im wesentlichen auf Funktionen mit dem Namen *Handle_AcMain_*.... zur Abhandlung von Kommandos an das Hauptfenster oder direkt auf Aufrufe der Form *Form_...* in die Formular-DLL abgebildet. Die Bedeutung der Funktionen der Formular-DLL wird in einem gesonderten Kapitel beschrieben.

Die Kommandos an das Hauptfenster werden in zwei Gruppen unterschieden:

Kommandos, die immer ausgeführt werden können
Kommandos, die sich auf das augenblicklich geöffnete Formular beziehen

Letztere Funktionen werden grundsätzlich mit dem Handle des Formularfensters versorgt, welcher zuvor aus dem aktiven MDI-Fenster durch Auswerten der angehefteten Struktur *MdiState* extrahiert wurde.

Die Routine Handle_MsgMain_FormMessage

Diese Funktion handelt alle Messages ab, die von der Formular-DLL an das Hauptfenster der Applikation gesandt werden.

Einige der Messages werden behandelt, damit in der Formularverarbeitung die Makrobefehle *Öffnen, NeueVorlage* oder *Schließen* korrekt durchgeführt werden. Diese sind:

FORMMSG_PERFORMNEW:
FORMMSG_PERFORMOPEN:
FORMMSG_PERFORMCLOSE:

Es muss der Handle des Formularfensters zurückgeliefert werden, welches *nach* der Ausführung der Message das aktive ist oder NULL, wenn es nach der Message kein offenes Formularfenster mehr gibt.

Näheres zu diesen, wie zu allen anderen, rein informativen, Message-Typen sind in einem gesonderten Kapitel über Messages beschrieben.

Die Callback-Funktion für MDI-Fenster

Die Callback-Funktion für MDI-Fenster wird durch folgenden Prozedurkopf eingeleitet:

```
CBFUNC (long) WndProcMdi
        (HWND hwnd, MSGPARAM message,
        WPARAM wParam, LPARAM lParam)
```

Das Makro CBFUNC (Typ) ist in der Headerdatei der Formularverwaltung definiert. Es wird verwendet, um die Quellhaltung syntaktisch für die 16- und 32-Bit-Welt kompatibel zu halten.

Alle weiteren Messages sind entweder Standard-Messages oder sollen hier nicht weiter diskutiert werden, da sie nichts mit der eigentlichen Formularverwaltung zu tun haben.

Die Routine Handle_MsgMdi_Create

Diese Routine handelt zentral alle Messages vom Typ *WM_CREATE* für MDI-Fenster ab.

Unter anderem wird in dieser Routine eine Kontrollstruktur *MdiState* vom Typ *MDISTATE* angelegt und an das MDI-Fenster angeheftet. In dieser Kontrollstruktur gibt es eine Variable *hwndForm.* Hier wird der Handle des Formularfensters abgelegt, welches in wenigen Augenblicken kreiert wird. Durch Lesen der Kontrollstruktur kann auf diese Weise der Handle des Formularfensters, das dem MDI-Fenster zugeordnet ist, jederzeit ermittelt werden.

Nachdem die Kontrollstruktur angelegt ist, wird der Aufruf an die Formular-DLL abgesetzt, welcher in der Client-Area des MDI-Fensters das Formularfenster eröffnet:

```
MdiState->hwndForm = Form_Open_Expanded
                     (hwnd, FileName, FileType, 0)
```

Falls der Aufruf fehlschlägt (Funktionswert 0 anstelle eines gültigen Fensterhandles), wird die Kontrollstruktur wieder gelöscht und die Kreation des MDI-Fensters abgebrochen.

Im Erfolgsfalle wird die Kontrollstruktur an das MDI-Fenster geheftet:

```
SetWindowLong (hwnd, 0, (long) MdiState);
```

Da die Funktion *Form_Open_Expanded* das Formular immer im *Ediermodus* eröffnet, muss, falls das Formular im Ausfüllmodus betrieben werden soll (wie in diesem Beispiel), sofort anschließend in den Ausfüllmodus umgeschaltet werden:

```
if (Form_SetWorkMode (MdiState->hwndForm, FORMMODE_RUN) < 0)
        return -1;
```

Es werden noch einige weitere Einstellungen am Formularfenster vorgenommen, wie z.B. das Setzen von speziellen Farbeinstellungen und des Vergrößerungsfaktors. Diese Einstellungen sind optional und applikationsabhängig.

Die Routine Handle_MsgMdi_Destroy

Diese Routine handelt zentral alle Messages vom Typ *WM_DESTROY* für MDI-Fenster ab.

In dieser Routine wird im wesentlichen die während der Message *WM_CREATE* angelegte Kontrollstruktur *MdiState* vom Fenster abgekoppelt und der zugehörige Speicher freigegeben.

Die Routine Handle_MsgMdi_Close

Diese Routine handelt zentral alle Messages vom Typ *WM_CLOSE* für MDI-Fenster ab.

Insbesondere wird hier die interne Merkvariable des dem MDI-Fenster zugeordneten Formulars abgefragt, die Auskunft darüber gibt, ob das Formular verändert wurde oder nicht. Dies geschieht mittels der Funktion:

```
if (! Form_GetChangeFlag (MdiState->hwndForm))
        return 1;
```

Wenn Änderungen stattgefunden haben, wird gefragt, ob diese gespeichert werden sollen. Im Falle einer Bejahung, wird das Formular mittels der Funktion:

```
if (Handle_AcForm_Save (MdiState->hwndForm) == IDCANCEL)
        return 0;
```

gespeichert, bevor das Fenster geschlossen wird.

Die Routine Handle_MsgMdi_FormMessage

Diese Routine handelt zentral alle Messages der Formular-DLL an das MDI-Fenster ab. Der Parameter *wParam* beschreibt den Typ der Aktion. Das Behandeln der Messages ist rein informativ und optional. Es braucht keine Message behandelt zu werden.

Unbehandelte Message-Typen sollten mit dem Return-Wert 0 beantwortet werden. Die Gesamtheit aller Message-Typen ist in einem gesonderten Kapitel beschrieben.

Die Routine Handle_AcMain_New

Diese Funktion wickelt das Kommando *Neue Vorlage* im Ausfüllprogramm der Formularverarbeitung ab. Es wird zunächst der Name der Vorlage beim Benutzer angefragt. Dann wird die

Kontrollstruktur *MdiCreateStruct* aufgebaut, welches das Windows-API für das Anlegen eines neuen MDI-Fensters benötigt.

Insbesondere wird der Name der Vorlage *FileName* auf der Variablen *lParam* der Kontrollstruktur abgelegt. Dieser Parameter wird später in der Abhandlung der Message *WM_CREATE* wieder ausgelesen und beim Anlegen des Formularfensters mittels der Funktion *Form_Open_Expanded* an die Formular-DLL weitergereicht.

Nach dem Öffnen des MDI-Fensters, welches implizit in der *WM_CREATE* Message auch das zugehörige Formularfenster eröffnet hat, wird die Kontrollstruktur *MdiState* aus dem Fenster ausgelesen, welche ebenfalls in der Behandlung der *WM_CREATE* Message an das Fenster angebunden wurde. Diese Kontrollstruktur beinhaltet den Fensterhandle des Formularfensters in der Variablen *hwndForm*.

Mittels dieses Handles wird das neu angelegte Formular sogleich angesprochen:

Form_MakeUntitled erklärt die neue Formular als *unbenannt*
Form_SetDefaults setzt die Vorbesetzung in alle Eingabeobjekte
Form_PerformBeginMacro führt das Makro *bei Beginn der Bearbeitung* aus
Form_SetChangeFlag setzt die interne Merkvariable *geändert* zurück

Prinzipiell kann ab jetzt jede Funktion der Formular-DLL aufgerufen werden. Eine ähnliche Abhandlung erfolgt, wenn ein Formular aus der mitgelieferten Formularbibliothek eröffnet wird. Die zugehörige interne Routine für diese Leistung lautet *Handle_AcMain_NewLibForm*.

Die Routine Handle_AcMain_Open

Ähnlich der Funktion *Handle_AcMain_New* eröffnet auch diese Funktion ein neues Formularfenster. Allerdings wird dieses nicht aus einer Vorlage eröffnet, die anschließend *unbenannt* ist, sondern es wird ein zuvor fertig ausgefülltes und abgespeichertes Formular wiedereröffnet.

Die Objekte dieses Formulars werden im Gegensatz zur Routine *Handle_AcMain_New* nicht vorbesetzt, da sie vom Benutzer ausgefüllte Werte enthalten.

Die Routine Handle_AcForm_Save

Diese Routine handelt das Speichern eines benannten oder unbenannten Formulars ab. Zunächst wird mittels der Funktion *Form_GetTitleFileName* geprüft, ob das Formular bereits benannt ist. Wenn nein, wird die Funktion *Handle_AcForm_SaveAs* aufgerufen, die für das Speichern unbenannter Formulare zuständig ist. Ansonsten wird das Formular mit der Funktion *Form_Save* in die bereits benannte Datei gespeichert.

8. Einbindung in ein Visual Basic Projekt

Dieses Kapitel soll dem Entwickler helfen, die Formular-DLL in eine Rahmenapplikation einzubinden, die in Visual Basic programmiert ist. Als Anschauungsobjekt wird das Visual Basic Projekt *formtest.vbp* verwendet. Das Projekt wurde in Visual Basic 5.0 verfaßt. Es wird somit nur die 32-Bit-Version unterstützt.

Der Quellcode des Projektes wird bei der Installation nebst allen benötigten Formularen und Modulen *vb* des Verzeichnisses kopiert, das als Ziel der Installation angegeben wurde. Das Projekt kann sofort aufgerufen und gestartet werden. Voraussetzung ist lediglich, dass alle benötigten dynamischen Bibliotheken im Verzeichnis *windows**system* zur Verfügung stehen.

Der Quellcode des Projektes darf modifiziert und als Basis für eigene Entwicklungen verwendet werden. Er ist frei von Lizenzen, wenn der Entwicklungskit der Formularsoftware ordnungsgemäß lizenziert wurde.

Bei dem Projekt handelt es sich um eine MDI-Applikation. Es können eines oder mehrere Formulare gleichzeitig geöffnet werden. Die Formulare können ausgefüllt und gedruckt werden
.

8.1. Aufrufkonventionen

Wie bereits eingangs dieser Dokumentation beschrieben, erwartet Visual Basic als Aufrufkonvention für externe DLL-Schnittstellen die Konvention *stdcall (Standard-Call).* Die Schnittstellen der Formularverwaltung werden sämtlich in beiden Möglichen Aufrufkonventionen (*cdecl* oder *stdcall*) angeboten. Sie unterscheiden sich durch ihre Namenskonvention. Die Schnittstellen der *cdecl*-Konvention beginnen alle mit dem Namens-Prefix *Form_*, während die Schnittstellen für die *stdcall*-Konvention mit dem Prefix *Sform_* beginnen. Die in dieser Dokumentation beschriebenen Schnittstellen sind alle in der *cdecl*-Namenskonvention ausgeführt. Da Visual Basic *stdcall* benutzt, müssen Sie daran denken, dem Namen der beschriebenen Schnittstelle jeweils ein großes *S* voranzustellen.

8.2. Deklaration aller Symbole und Funktionen

Wenn in Visual Basic Funktionen oder Subroutines einer externen DLL aufgerufen werden sollen, müssen diese in einem statischen Modul namentlich und mit dem Typ aller Parameter deklariert sein. Dies geschieht mit dem Befehl

Declare Function ... oder *Declare Sub*

Wenn zum Beispiel die Routine *Form_Open_Expanded* aufgerufen werden soll, muss diese wie folgt deklariert werden:

Declare Function *SForm_Open_Expanded* Lib "*formdl32.dll*"
(ByVal *HwndHost* As Long, ByVal *FormFile* As String,
ByVal *FileType* As Integer, ByVal *Flags* As Long) As Long

Damit Sie nicht alle in der Formularsoftware bereitgestellten Schnittstellen selbst deklarieren müssen. haben wir das bereits für Sie vorbereitet. Im Verzeichnis *vb* wird eine Datei *formdll.bas* bereitgestellt, die bereits alle Schnittstellen fertig deklariert enthält.

Darüberhinaus werden in dieser Datei auch alle in der Beschreibung benutzten Konstanten als *Public Const* Anweisung deklariert. Zum Beispiel:

Public Const *FORMMODE_RUN* as integer = 1
Public Const *FORMMODE_EDIT* as integer = 2

Damit stehen dem Visual Basic Programmierer dieselben Konstantendefinitionen zur Verfügung, die dem C- oder C++ Programmierer in der Definitionsdatei *formdll.h* angeboten werden.

Die Datei *formdll.bas* braucht lediglich als Modul in die fertige Visual Basic Applikation eingebunden zu werden.

8.3. Transfer von Zeichenketten

Für den Transfer von Zeichenketten als Parameter Funktion in einer externen DLL sind bei Visual Basic einige Besonderheiten zu beachten. Dies liegt darin begründet, dass Zeichenketten (Strings) in Visual Basic als eigenständige Objekte von variabler Größe verwaltet werden, während sie in C oder C++ (die Sprache der DLL-Funktionen) als Seicherbereiche definiert sind, in denen die Zeichenfolge mit dem ersten Nullbyte endet.

Dadurch ergeben sich folgende Besonderheiten für den Transfer von Strings aus Visual Basic in eine externe Routine oder umgekehrt.

Strings an eine DLL-Funktion übergeben

Diese Richtung ist bemerkenswert einfach. Sie brauchen lediglich den Parameter in der Deklaration der externen Funktion mit den Attributen *byval as String* zu versehen und die Zeichen

folge als String-Variable oder String-Konstante als Parameter angeben. Der String erreicht die externe Routine als korrekter C-String.

Diese Anwendung findet sich zum Beispiel, wenn mittels der Funktion *FormObj_SetText* der Inhalt eines Eingabefeldes eines Formulars mit einem bestimmten Text gefüllt werden soll. Der Aufruf lautet einfach:

```
FormObj_SetText   Handle, "Der Neue Text", 0, 0
```

Strings aus einer DLL-Funktion übernehmen

Diese Richtung ist etwas aufwendiger. Soll ein Visual Basic String mittels einer externen Funktion gefüllt werden, muss zunächst ein String hinreichender Länge erzeugt werden, sagen wir z.B. 1000 Zeichen. Dies geschieht mittels der Anweisung *space*:

```
x = space (1000)
```

Dann wird die externe Routine aufgerufen und der String *x* als Parameter angegeben. In der Regel haben alle Routinen, die einen String zurückliefern auch einen Parameter, der der Routine die Maximallänge des Resultat-Strings mitteilt. In dem angegebenen Beispiel sollte dann als Maximallänge der Wert 999 angegeben werden, also ein Zeichen weniger als der String in Visual Basic lang ist. Das letzte Zeichen wird von der C-Funktion für das abschließende Nullbyte benötigt.

Nach dem Aufruf der Funktion beinhaltet der Visual Basic String das Ergebnis - jedoch leider in Form eines C-Strings, also mit einem Nullbyte als Begrenzer und in der vollen Länge (also hier 1000). Das Ergebnis muss jetzt noch in einen Basic-String umgewandelt werden. Dazu ist zunächst die Position des Nullbytes mittels der Basic-Funktion *instr* auszumachen. Anschließend muss dann das Nullbyte und der Rest des Strings mit der Funktion *left* abgeschnitten werden.

Die folgende Sequenz legt den Inhalt des Eingabeobjektes mit dem Handle *hFob* auf dem Basic-String *Ergebnis* ab:

```
Ergebnis = space (1000)
FormObj_GetText  hFob, Ergebnis, 999
Ergebnis = left (Ergebnis, instr (Ergebnis, chr (0)))
```

Am besten verpacken Sie die ganze Vorgehensweise in ein Unterprogramm und verwenden an dieser Stelle immer das Unterprogramm.

Anhang OBJEKTTYPEN

Liste aller *[e] forms & more* Objekttypen

Objekttyp	Objektart	Bedeutung
FOTYPE_TEXT	Gestaltung	Textobjekt (einzeilig oder mehrzeilig)
FOTYPE_RTF	Gestaltung	Ausgabe von RTF-Texten mit Formatierungen
FOTYPE_RECT	Gestaltung	Rechteck
FOTYPE_LINE	Gestaltung	Linie
FOTYPE_CIRCLE	Gestaltung	Kreis oder Ellipse
FOTYPE_PICTURE	Gestaltung	Bild (Bitmap, Jpeg)
FOTYPE_DATE	Einblendung	Aktuelles Datum
FOTYPE_TIME	Einblendung	Aktuelle Uhrzeit
FOTYPE_COPYNAME	Einblendung	Name der Ausfertigung
FOTYPE_EDIT	Eingabe	Eingabefeld (einzeilig oder mehrzeilig)
FOTYPE_COMBO	Eingabe	Auswahlfeld (Auswahl, wahlweise auch Eingabe)
FOTYPE_CHECKBOX	Eingabe	Ankreuzfeld (auslösend oder nicht auslösend)
FOTYPE_BUTTON	Eingabe	Schaltfläche
FOTYPE_TABLE	Eingabe	Tabelle
FOTYPE_ODBCACC	Datenzugriff	Zugriff auf ODBC Tabellen

Kombinationen von Objekttypen

Kombination	Bedeutung
FOTYPE_ALL	Bitliste aller Objekttypen
FOTYPE_INPUT	Bitliste aller Eingabeobjekttypen
FOTYPE_ACCESS	Bitliste aller Datenzugriffsobjekttypen

Anhang NOTIFICATIONS

Die Kernbibliothek von [e] forms & more sendet bei allen erdenklichen Ereignissen Nachrichten an das Host-Fenster – sogenannte *Notification Messages* oder kurz *Notifications.*

Die meisten *Notifications* haben rein informativen Charakter. Die Rahmenapplikation kann diese Informationen auswerten oder ignorieren.

Manche Messages sind allerdings als Anfragen des Formulars an die Rahmenapplikation zu verstehen und müssen bzw. sollten beantwortet werden.

Die Messages werden je nach Bedeutung an einen dieser beiden Empfänger versandt:

- Host-Fenster eines Formulars
- Hauptfenster der Rahmenapplikation

Die Rahmenapplikation muss zu Beginn einen eindeutigen Message-Typ registrieren, um die Kommunikation zwischen dem Formular und Rahmenapplikation zu gewährleisten. Dies geschieht durch den Aufruf:

```
Global.FormMessage = RegisterWindowMessage (MESSAGE_FORM);
```

Die *Notifications* werden als *Windows-Messages* zugestellt. *Windows Messages* haben grundsätzlich zwei Parameter. Diese haben folgende Bedeutung:

wParam	Typ der Notification (siehe nachstehende Tabelle)
lParam	Spezifische Zusatzinformation (abhängig von der Notification)

Die folgenden Tabellen erläutern alle Notifications, die von *[e] forms & more* versendet werden.

Notification Messages an das Host-Fenster

Notification	Ursache
FORMMSG_CREATEFORM	Es wurde ein neues Formularfenster eröffnet.
FORMMSG_DESTROYFORM	Das Formularfenster wird geschlossen.
FORMMSG_ADDPAGE	Es wurde eine neue Seite angelegt.
FORMMSG_DELETEPAGE	Es wurde eine Seite gelöscht.
FORMMSG_EDITPAGE	Die Kenndaten einer oder mehrerer Seiten wurden verändert.
FORMMSG_CURRENTPAGE	Die *aktuelle* Seite hat gewechselt. Es ist jetzt eine andere Seite aktiv als vorher.
FORMMSG_ADDOBJECT	Es wurde ein neues Objekt angelegt.
FORMMSG_ADD_PAGEZONE	Es wurde eine neue Seitenzone angelegt.
FORMMSG_DELETEOBJECT	Es wurde ein Objekt gelöscht.
FORMMSG_DELETE_PAGEZONE	Eine Seitenzone wurde gelöscht.
FORMMSG_OBJECT_CONTENT_CHANGE	Der Inhalt eines Objekts hat sich zur Laufzeit geändert. Der Handle des Objekts (HFOB) wird in *lParam* übergeben.
FORMMSG_EDITOBJECT	Die Kenndaten eines Objekts wurden verändert.
FORMMSG_EDIT_PAGEZONE	Die Kenndaten einer Seitenzone wurden verändert.
FORMMSG_SELECTOBJECT	Es wurde ein Objekt markiert oder die Markierung eines Objekts wurde entfernt.
FORMMSG_SELECT_PAGEZONE	Eine Seitenzone wurde markiert oder die Markierung wurde entfernt.
FORMMSG_MOVEORSIZEOBJECT	Die Größe oder die Position eines Objekts wurde verändert.
FORMMSG_MOVEORSIZE_PAGEZONE	Die Größe oder die Position einer Seitenzone wurde verändert.

Notification	**Ursache**
FORMMSG_ZOOMCHANGE	Der Vergrößerungsfaktor (Zoom) der Anzeige wurde verändert.
FORMMSG_GRIDCHANGE	Das Raster wurde verändert.
FORMMSG_COLORMODECHANGE	Der Schwarz/Weiß-Modus wurde ein- oder ausgeschaltet.
FORMMSG_STARTMOVEMODE	Es wurde begonnen, ein Objekt mit der Maus zu bewegen. Die Message wird beim Drücken der Maustaste gesendet. *lParam* zeigt auf ein Rechteck welches die augenblickliche Position und Größe des Objekts enthält.
FORMMSG_UPDATEMOVEMODE	Es wird eine neue Zwischenposition beim Verschieben eines Objekts mit der Maus vermeldet. Der Vorgang ist noch nicht abgeschlossen. Die Maustaste ist immer noch gedrückt. *lParam* zeigt auf ein Rechteck, welches die augenblickliche Position und Größe des Objekts enthält.
FORMMSG_ENDMOVEMODE	Das Bewegen eines Objekts mit der Maus wurde abgeschlossen. Die Message wird versendet, wenn die Maustaste losgelassen wird.
FORMMSG_STARTSIZEMODE	Es wurde begonnen, ein Objekt mit der Maus zu vergrößern oder zu verkleinern. Die Message wird beim Drücken der Maustaste gesendet. *lParam* zeigt auf ein Rechteck, welches die augenblickliche Position und Größe des Objekts enthält.
FORMMSG_UPDATESIZEMODE	Es wird eine neue Zwischenposition beim Vergrößern oder Verkleinern eines Objekts mit der Maus vermeldet. Der Vorgang ist noch nicht abgeschlossen. Die Maustaste ist immer noch gedrückt. *lParam* zeigt auf ein Rechteck, welches die augenblickliche Position und Größe des Objekts enthält.

Notification	**Ursache**
FORMMSG_ENDSIZEMODE	Das Vergrößern oder Verkleinern eines Objekts mit der Maus wurde abgeschlossen. Die Message wird versendet, wenn die Maustaste losgelassen wird.
FORMMSG_BEFORECREATE PRINTERDC	Diese Message wird unmittelbar vor dem Kreieren eines Printer-DC versendet. Der Parameter *lParam* zeigt auf die *DEVMODE*-Struktur, mit der der *Printer-DC* eröffnet wird. Die Applikation kann auf diese Weise vor dem Kreieren des Device-Kontexts noch Einfluss auf die *DEVMODE*-Struktur nehmen.
FORMMSG_SETFOCUS	Ein anderes Formular oder ein anderes Formularobjekt erhält den Fokus. Der Handle des Objektes, das den Fokus erhält wird in *lParam* übergeben.
FORMMSG_KILLFOCUS	Ein Objekt hat den Fokus verloren. Der Handle des Objektes, das den Fokus verliert, wird in *lParam* übergeben.
FORMMSG_RUNCHANGE	Das Formular (*Ediermodus)* oder der Inhalt eines Objekts (*Ausfüllmodus*) wurde verändert.
FORMMSG_EDITORDERCHANGE	Die Reihenfolge der Eingabeobjekte wurde verändert.
FORMMSG_WORKMODECHANGE	Es wurde vom *Ausfüllmodus* in den *Ediermodus* oder umgekehrt umgeschaltet.
FORMMSG_F2PREFIX_KEY	Es wurde im eingeschalteten Kommando-Modus eine Taste betätigt. Der Wert der Taste wir auf *lParam* übergeben. Einige Applikationen wie z.B. der Formulareditor interpretieren die Tastenfolge F2+‘x‘ als Kommando. Da einige Objekte wie z.B. das Eigabefeld und die Combobox selbst alle Key-down-Messages abhandeln, wird diese Nachricht an das Hostfenster geschickt.

Notification	**Ursache**
FORMMSG_F2PREFIX_TOGGLE	Es wurde der Status des Kommando-Modus gwechselt (siehe FORMMSG_F2PREFIX_KEY).
FORMMSG_UNACCESS	Es wurde die aktuelle Selektion aufgehoben. Es befindet sich kein Datensatz mehr um Zugriff. Diese Nachricht dient z.B. zum Update der Statuszeile.
FORMMSG_RECORDACCESS	Es wurde ein Datenzugriff folgender Art durchgeführt: Ein neues Recordset wurde selektiert. Es wurde auf einen anderen Datensatz positioniert. Diese Nachricht dient z.B. zum Update der Statuszeile.
FORMMSG_RECORDUPDATE	Es wurde der aktuelle Datensatz in die Datenbank zurückgeschrieben.
FORMMSG_RECORDINSERT	Es wurde ein neuer Datensatz in die Datenbank eingefügt.
FORMMSG_RECORDDELETE	Es wurde ein Datensatz in der Datenbank gelöscht.
FORMMSG_WHERE_CLAUSE_CHANGED	Das aktuelle Selektionskriterium (*SQL: Where Clause*) hat sich geändert. Ein Zeiger auf eine Zeichenkette, die das neue Kriterium beinhaltet, wird auf *lParam* überliefert.
FORMMSG_ORDERBY_CLAUSE_CHANGED	Das aktuelle Sortierkriterium (*SQL: OrderBy Clause*) hat sich geändert. Ein Zeiger auf eine Zeichenkette, die das neue Kriterium beinhaltet, wird auf *lParam* überliefert.
FORMMSG_MATCHMODE_CHANGED	Der Status der Suchbegriffeingabe hat sich geändert. Der aktuelle Status wird auf *lParam* übergeben. 0 Suchbegriffeingabe ist aus. 1 Suchbegriffeingabe ist aktiv.

Notification	**Ursache**
FORMMSG_START_BROADCAST FORMMSG_OBJECT_BROADCAST FORMMSG_END_BROADCAST	Diese drei Nachrichten werden bei Ausführung der Makrofunktion *BroadcastAllInputs* gesendet. Es werden der Name und der Inhalt aller Eingabefelder an ein im Makro spezifiziertes Fenster verschickt. Die Message *FORMMSG_START_BROADCAST* leitet das Geschehen ein. Die Message *FORMMSG_END_BROADCAST* beendet es. Beide Messages sind parameterlos. *FORMMSG_OBJECT_BROADCAST* wird für jedes Eingabeobjekt verschickt. **Auf *lParam*** wird ein Zeiger auf eine Struktur des Typs *FORM_OBJECT_BROADCAST* übergeben, welche Aufschluss über Namen und Inhalt des Objektes gibt.
FORMMSG_ASK_FULL_HTML_LIZENSE	Mit dieser Message fragt die Formular-DLL in einigen Situationen an, ob wirklich eine volle HTML-Exportlizenz vorliegt. Dies ist nicht zu verwechseln mit der Nachricht FORMMSG_REGREQUEST_HTML_EXPORT, da dort möglicherweise auch in einer Demo-Version ein HTML-Export erlaubt wird. **!= x HTML-Exportlizenz liegt nicht vor.** **== x HTML-Exportlizenz liegt vor.** Der Returnwert für den HTML-Export (x) kann bei der Firma **Waimea Software GmbH** erfragt werden.

Notification	**Ursache**
FORMMSG_LBUTTONDOWN FORMMSG_LBUTTONUP FORMMSG_RBUTTONDOWN FORMMSG_RBUTTONUP	Der Benutzer hat die linke oder rechte Maustaste im Formularfenster gedrückt. Diese Nachrichten werden sowohl im Arbeitsmodus *FORMMODE_EDIT* als auch im Modus *FORMMODE_RUN* gesendet. Auf dem Parameter *lParam* wird die Position des Mauszeigers übergeben. Diese wird relativ zum Formularfenster in Pixel angegeben. In den höherwertigen 2 Bytes ist dir vertikale Kooridinate, in den niederwertigen 2 Bytes die horizontale Koordinate abgelegt. Diese Koordinaten können den Routinen *Form_GetPointedObject* und *Form_PixelToPos* direkt zur Weiterverarbeitung als Parameter mitgegeben werden.
FORMMSG_HOTKEY	Es wurde eine Taste oder Tastenkombination gedrückt, für die mittels der API-Funktion *Form_WantHotKey* angemeldet wurde, dass diese als Notifikation an das Host-Fenster gesendet wird. In Parameter *lParam* wird die Taste und die ggf. ebenfalls gedrückten Zusatztasten übergeben. Bit 0 – 15 (Maske 0x0000FFFF) Wert der Taste Bit 16 (Maske 0x00010000) 1 wenn Umschalt-Taste gedrückt, sonst 0 Bit 17 (Maske 0x00020000) 1 wenn Strg-Taste gedrückt, sonst 0

Notification Messages an das Rahmenfenster der Anwendung

Notification	Ursache
FORMMSG_PERFORMOPEN	Diese Message wird gesendet, wenn der Makrointerpreter den Makrobefehl *NeueVorlage* ausführen soll. Diese Leistung delegiert er an die Rahmenapplikation. Auf dem Parameter *lParam* wird die Adresse eines Strings mit dem Namen der Formularvorlage übergeben. Als Rückgabewert der Message übergibt die Rahmenapplikation den Handle des nach dem Öffnen aktiven Formularfensters. Dies ist das neu eröffnete Fenster, wenn kein Fehler aufgetreten ist.
FORMMSG_PERFORMNEW	Diese Message wird gesendet, wenn der Makrointerpreter den Makrobefehl *Öffnen* ausführen soll. Diese Leistung delegiert er an die Rahmenapplikation. Auf dem Parameter *lParam* wird die Adresse eines Strings mit dem Namen des zu öffnenden Formulars übergeben. Als Return-Wert der Message übergibt die Rahmenapplikation den Handle des nach dem Öffnen aktiven Formularfensters. Dies ist das neu eröffnete Fenster, wenn kein Fehler aufgetreten ist.
FORMMSG_PERFORMCLOSE	Diese Message wird gesendet, wenn der Makrointerpreter den Makrobefehl *Schließen* ausführen soll. Diese Leistung delegiert er an die Rahmenapplikation. Diese schließt ein Formularfenster ihrer Wahl – in der Regel das augenblicklich aktive. Als Return-Wert der Message übergibt die Rahmenapplikation den Handle des nach dem Schließen aktiven Formularfensters. Wenn keine Formularfenster mehr offen sind, wird der Wert 0 übergeben.
FORMMSG_PERFORMQUIT	Diese Message wird gesendet, wenn der Makrointerpreter den Makrobefehl *Beenden* ausführen soll. Diese Leistung delegiert er an die Rahmenapplikation. Diese beendet in der Regel die Ausführung des Programms.

Anhang VERALTETE FUNKTIONEN

Folgende veraltete Funktionen werden zwar (noch) von *[e] forms & more* bereitgestellt, sollten jedoch nicht mehr verwendet werden, da ein Support dieser Funktionen in Zukunft nicht gewährleistet werden kann.

Formularfunktionen

Veraltete Funktion	Man verwende anstelle dessen
Form_AdjustSizeOfObjects	Form_ArrangeObjects
Form_AlignObjects	Form_ArrangeObjects
Form_ArrangeSelectedObjects	Form_ArrangeObjects
Form_ChangeObjectType	Form_ChangeObjectsType
Form_ChangePrimaryObjectType	Form_ChangeObjectsType
Form_CopySelectedObjects	Form_ObjectApi
Form_CopyTextFromPrimarySelectedObject	Form_ObjectApi
Form_CountObjects	Form_ObjectApi
Form_CreateNewObject	Form_CreateObject
Form_CreateNewObjectExtended	Form_CreateObject
Form_CreateNewPage	Form_PageApi
Form_CutSelectedObjects	Form_ObjectApi
Form_DeleteCurrentPage	Form_PageApi
Form_DeleteObjectsByName	Form_ObjectApi
Form_DeleteSelectedObjects	Form_ObjectApi
Form_DuplicateObjects	Form_DuplicateObjectsExt
Form_EditAlignMode	--- kein Ersatz ---
Form_EditArrangeMode	--- kein Ersatz ---
Form_EditColors	Form_EditObjects
Form_EditConfig	Form_EditFormProperties

Veraltete Funktion	Man verwende anstelle dessen
Form_EditCopyList	Form_EditFormProperties
Form_EditCurrentPage	Form_PageApi
Form_EditDisplayOptions	Form_EditFormProperties
Form_EditFontColors	Form_EditObjects
Form_EditFonts	Form_EditObjects
Form_EditFormMacros	Form_EditFormProperties
Form_EditFrameColors	Form_EditObjects
Form_EditGrid	Form_EditFormProperties
Form_EditHtmlExport	Form_EditFormProperties
Form_EditObjectCollection	Form_EditObjectCollectionExt
Form_EditObjectProperties	Form_EditObjects
Form_EditOptions	Form_EditFormProperties
Form_EditOutlines	Form_EditObjects
Form_EditPasswords	Form_EditFormProperties
Form_EditPdfExport	Form_EditFormProperties
Form_EditPrimarySelectedObject	Form_EditPrimarySelectedObjectExt
Form_EditPrintExpiration	Form_EditFormProperties
Form_EditPrintParams	Form_EditFormProperties
Form_EditProperties	Form_EditObjectProperties
Form_EditRaster	Form_EditFormProperties
Form_EditSameSpaceMode	--- kein Ersatz ---
Form_EditSizes	Form_EditObjects
Form_GetExtraColor	Form_GetColorProperty
Form_GetPageInfo	Form_GetLongProperty
Form_GetStringProperty	Form_GetXmlProperty
Form_ObjectsApi	Form_ObjectApi
Form_Open	Form_Open_Expanded

Veraltete Funktion	**Man verwende anstelle dessen**
Form_PasteObjects	Form_ObjectApi
Form_PasteObjectsToPage	Form_ObjectApi
Form_PasteTextToObjects	Form_ObjectApi
Form_PasteTextToSelectedObjects	Form_ObjectApi
Form_SelectedObjectsToBackground	Form_ObjectApi
Form_SelectedObjectsToForground	Form_ObjectApi
Form_SetExtraColor	Form_SetColorProperty
Form_SetLtsLanguageCode	--- kein Ersatz ---
Form_SetObjectNamesPool	Form_SetXmlPropertyByAttrName
Form_SetWorkMode	Form_SetWorkModeExt

Seitenfunktionen

Veraltete Funktion	Man verwende anstelle dessen
FormPage_GetLongProperty	FormPage_GetXmlPropertyByAttrName
FormPage_GetStringProperty	FormPage_GetXmlPropertyByAttrName
FormPage_SetLongProperty	FormPage_SetXmlPropertyByAttrName
FormPage_SetStringProperty	FormPage_SetXmlPropertyByAttrName

Objektfunktionen

Veraltete Funktion	Man verwende anstelle dessen
FormObj_GetColorProperty	FormObj_GetXmlPropertyByAttrName
FormObj_GetComboList	FormObj_GetXmlPropertyByAttrName
FormObj_GetDataField	FormObj_GetXmlPropertyByAttrName
FormObj_GetDataType	FormObj_GetXmlPropertyByAttrName
FormObj_GetLongComment	FormObj_GetXmlPropertyByAttrName
FormObj_GetLongCommentLng	--- kein Ersatz ---
FormObj_GetLongProperty	FormObj_GetXmlPropertyByAttrName
FormObj_GetName	FormObj_GetXmlPropertyByAttrName
FormObj_GetOutline	FormObj_GetXmlPropertyByAttrName
FormObj_GetShortComment	FormObj_GetXmlPropertyByAttrName
FormObj_GetStringProperty	FormObj_GetXmlPropertyByAttrName
FormObj_GetText	FormObj_GetXmlPropertyByAttrName
FormObj_GetTypeName	FormObj_GetXmlPropertyByAttrName
FormObj_GetType	FormObj_GetXmlPropertyByAttrName
FormObj_ResetStateBits	FormObj_GetState, FormObj_SetState
FormObj_SetColorProperty	FormObj_SetXmlPropertyByAttrName
FormObj_SetComboList	FormObj_SetXmlPropertyByAttrName
FormObj_SetLongProperty	FormObj_SetXmlPropertyByAttrName
FormObj_SetOutline	FormObj_SetXmlPropertyByAttrName

Veraltete Funktion	**Man verwende anstelle dessen**
FormObj_SetStateBits	FormObj_GetState, FormObj_SetState
FormObj_SetStringProperty	FormObj_SetXmlPropertyByAttrName
FormObj_TestStateBits	FormObj_GetState
FormObj_ToggleStateBits	FormObj_GetState, FormObj_SetState

Anhang FORM XML PROPERTIES

ActiveInputDisplay	1 = aktives Eingabefeld hervorheben 0 = aktives Eingabefeld nicht hervorheben
AutoZoomMode	0 = kein automatisches Zoomen 1 = Zoomen auf Fensterbreite 2 = Zoomen auf Fensterhöhe
ColorActiveInput	Farbe, in der das aktive Eingabefeld dargestellt wird. Die Farbe wird nur verwendet, wenn die Eigenschaft *ActiveInputDisplay* den Wert 1 hat.
ColorCommentAvail	Farbe, in der Eingabeobjekte dargestellt werden, für die ein Kommentar verfügbar ist
ColorMode	0 = Darstellung in Farbe 1 = Darstellung Schwarz-Weiss
ColorPageShadow	Farbe, in der der Schatten einer Seite dargestellt wird in der Form R,G,B
ColorWindowBgr	Farbe des Fensterhintergrundes in der Form R,G,B
DisableWarnings	0 = Warnungen immer anzeigen 1 = Warnungen nicht anzeigen
DisabledComponents	Bitliste, die einzelne Komponenten der Software ausschaltet. Folgende Bits können gesetzt werden: **Wert Bedeutung** 1 keine Makroeingabe 2 keine Eingabe von Javaskript für HTML 4 keine Eingabe von Javaskript für PDF 8 keine Datenzugriffsobjekte
DispDistBottom	Unterer Rand zwischen Fenster und Formularseite in Pixeln
DispDistHori	Seilicher (linker und rechter) Rand zwischen Fenster und Formularseite in Pixeln
DispDistInterpage	Abstand zwischen zwei Formularseiten in der Anzeige in Pixeln

DispDistTop	Oberer Rand zwischen Fenster und Formularseite in Pixeln
ExplicitTitle	Benutzerdefinierter Titel des Fensters
Grid	Raster für das Einraster von Objekten in Mikrometern
HelpMode	0 = Die Hilfefunktionalität ist ausgeschaltet. 1 = Die Standard-Hilfe der Windows-Umgebung wird verwendet. Diese Option ist für Fremdapplikationen gedacht, welche eine eigene Hilfe für das Gesamtprodukt implementieren. 2 = Testmodus
ModalDialogFlags	Bitliste, die den Betrieb des Formulars als modaler Dialog steuern. Folgende Bits können gesetzt werden: **Wert Bedeutung** 1 Dialog maximieren
ModalDialogHeight	Höhe des Dialogfensters in Pixeln bei der Verwendung des Formulars als modaler Dialog
ModalDialogWidth	Breite des Dialogfensters in Pixeln bei der Verwendung des Formulars als modaler Dialog
NextGroupId	Nächste zu vergebende ID beim Anlegen von Gruppen
NoSaveQuestion	0 = Beim Schließen fragen, ob gespeichert werden soll 1 = Beim Schließen nicht fragen, ob gespeichert werden soll
ObjectNamesPool	Namensvorrat für Objektnamen. Dieser wird in den Namensfeldern der Objekte als Wertevorrat angeboten. Die Liste ist durch Kommata separiert
PageKeyMode	Bestimmt das Verhalten der Tasten *Bild auf* und *Bild ab:* 1 = Blättern der Formularseite 2 = Datensatz *vor* und *zurück* beim Datenbankzugriff
RasterColor	Farbe des Gitternetzes in der Form R,G,B. Das Gitternetz wird nur angezeigt, wenn *RasterDistance* einen positiven Wert hat.

RasterDistance	Abstand der Gitternetzlinien in Mikrometern.
RasterZOrder	1 = Gitternetz im Vordergrund anzeigen 2 = Gitternetz im Hintergrund anzeigen
ReadOnlyMode	0 = Normaler Betrieb im Modus *Ausfüllen* 1 = Formular darf im Modus *Ausfüllen* nicht verändert werden
XmlSchema	Name der XML-Schemadatei des Objekt-Assistenten

Anhang PRINT XML PROPERTIES

BlackWhite	"0" Normaldruck "1" Druck in schwarz-weiß
NoBgrPicture	"0" Hintergrundbild normal drucken "1" Hintergrundbild nicht drucken
OnlyContent	"0" Normaldruck "1" Nur die Inhalte der Eingabefelder drucken
Duplex	"0" Normaldruck (einseitig) "1" Duplex-Druck (doppelseitig)
SerialMode	"0" Nur ein Formular drucken "1" Serienformular basierend auf Datenquelle drucken
Copies	"n" Anzahl der zu druckenden Kopien
Zoom	"n“ Druck-Zoom-Faktor in Prozent
PageMode	"0" Alle Seiten Drucken "1" Seitenbereich *FromPage* bis *ToPage* drucken "2" Nur die aktuelle Seite drucken
FromPage	"n" Erste zu druckende Seite
ToPage	"n" Letzte zu druckende Seite
OffsetX	"x" Horizontaler Druck-Offset in Mikrometern
OffsetY	"y" Vertikaler Druck-Offset in Mikrometern
CopyListMode	"0" Nur eine Ausfertigung drucken "1" Alle Ausfertigungen drucken, die in der durch das Attribut "CopyList" definierten Ausfertigungsliste definiert sind
OutputFile	"" Leere Zeichenfolge: Druckausgabe erfolgt auf den Drucker "file" Die Druckausgabe erfolgt in die Datei *file*

Anhang PAGE XML PROPERTIES

BgrColor	Farbe des Seitenhintergrundes in der Form R,G,B. Die Farbe ist nur relevant, wenn die Eigenschaft *BgrMode* den Wert 2 hat.
BgrMode	1 = Die Seite hat einen transparenten Hintergrund. 2 = Die Seite hat einen einfarbigen Hintergrund. Der Farbton kann als Farbeigenschaft *BgrColor* gesetzt werden. 3 = Im Hintergrund der Seite wird ein Bild eingeblendet
BgrPrint	1 = Hintergrund der Seite drucken 0 = Hintergrund der Seite nicht drucken
Height	Höhe der Seite in Mikrometern
Name	Name der Seite
Number	Nummer der Seite
Orient	1 = Hochformat 2 = Querformat
PicFrame	Rahmen des Hintergrundbildes in der Form: L,O,R,U. L = linker Rand O = oberer Rand R = rechter Rand U = unterer Rand Alle Angaben sind in Mikrometern relativ zum Seitenrand
PicHoriAlign	Horizontale Ausrichtung des Hintergrundbildes im Rahmen, falls das Bild nicht automatisch auf die Größe des Rahmens angepasst wird: 1 = linksbündig 2 = mittig 3 = rechtsbündig

PicSizeMode	Modus, wir das Bild in den Rahmen eingepasst wird: 1 = Bild in Rahmen einpassen, Seitenverhältnisse nicht beibehalten 2 = Bild in Rahmen einpassen, Seitenverhältnisse beibehalten 3 = Bild in Originalgröße darstellen 4 = Bild vergrößern um Zoomfaktor *PicZoom*
PicVertAlign	Vertikale Ausrichtung des Hintergrundbildes im Rahmen, falls das Bild nicht automatisch auf die Größe des Rahmens angepasst wird: 4 = oben ausrichten 2 = mittig 3 = unten
PicZoom	Zoomfaktor für die Vergrößerung des Hintergrundbildes. Ist nur relevant, wenn *PicSizeMode = 4* ist.
PrintFlag	1 = Seite wird gedruckt 0 = Seite wird nicht gedruckt
PrinterBin	Nummer des Druckerschachtes auf dem die Seite ausgegeben wird
Width	Breite der Seite in Mikrometern

Anhang OBJECT XML PROPERTIES

Eigenschaft	Objekttypen	Bedeutung
AccObject	Alle	Name des mit dem Objekt verbundenen Datenzugriffsobjekts
AccessFlags	Datenzugriffs-Objekte	Bitliste, die Optionen für Datenzugriffsobjekte spezifiziert, als dezimaler Wert. 1 Beim öffnen alle Datensätze selektieren 2 Beim Öffnen automatisch Matchmode ein 16 Aktionsanzeige *Selektieren* aktivieren 32 Aktionsanzeige *Schreiben* aktivieren 64 Aktionsanzeige *Löschen* aktivieren 128 Aktionsanzeige *Einfügen* aktivieren 256 Aktionsanzeige *Unselect* aktivieren 512 Bestätigung beim *Schreiben* aktivieren 1024 Bestätigung beim *Löschen* aktivieren 2048 Bestätigung beim *Einfügen* aktivieren 4096 3D-Rahmen 8192 Daten trimmen (Blanks am Ende entfernen) Die Werte sind in *formdll.h* als Konstanten definiert. Sie beginnen mit dem Prefix: *ACCOBJFLAG*

Eigenschaft	Objekttypen	Bedeutung
Action	Buttons	Aktion, die ein Button ausführt: 1 Makro abspielen 2 Formular zurücksetzen (auf Defaults) 3 Formular absenden 4 Javascript abspielen (nur HTML und PDF) 5 Hilfefunktion aufrufen Die Werte sind in *formdll.h* als Konstanten definiert FORM_BUTTACT_MACRO 1 FORM_BUTTACT_RESET 2 FORM_BUTTACT_SUBMIT 3 FORM_BUTTACT_JAVA 4 FORM_BUTTACT_HELP 5
AlignHori	Alle	Horizontale Ausrichtung des Objektinhalts als String: "left" "center" "right"
AlignVert	Alle	Vertikale Ausrichtung des Objektinhalts als String: "top" "center" "bottom"
BarrTag	Alle	Index des Barriere-Tags für den PDF-Export
ButtonGap		
ButtonSize		
CheckBoxFlags	Checkboxen	Bitliste, die Optionen für Checkboxen spezifiziert, in hexadezimaler Form: 0x00000000. Die Bits haben folgende Bedeutung: 0x01 Checkbox löst gegenseitig aus (Radio) 0x02 Label auf der linken Seite

Eigenschaft	Objekttypen	Bedeutung
CheckText	Checkboxen	Wert, der mit der Checkbox im angekreuzten Zustand assoziiert ist als Text
ColorBgr	Alle	Farbe des Hintergrunds in der Form R,G,B. Nur relevant, wenn das Flag *Transparent* nicht aktiv ist.
ColorDark	Buttons	Farbe des dunklen 3D-Schattens in der Form R,G,B
ColorFrame	Alle	Farbe des Randes in der Form R,G,B. Nur relevant, wenn der Rand eine positive Dicke hat.
ColorInnerFrame	Rechtecke	Farbe des inneren Randes in der Form R,G,B
ColorLight	Buttons	Farbe des hellen 3D-Schattens in der Form R,G,B
ComboList	Comboboxen	Auswahlliste einer Combobox. Die einzelnen Einträge sind durch LF (hex 0A) voneinander separiert.
ComboType	Comboboxen	Typ der Combobox: 1 = Nur Auswahl vorhandener Einträge 2 = Auswahl eines Eintrags oder manuelle Eingabe
Comment	Eingabeobjekte	Ausführlicher Kommentar zu einem Eingabefeld
Corner	Rechtecke	Eckenform (ganze Zahl): 1 = eckig 2 = mittel 3 = rund
CrossColor	Checkboxen	Farbe des Kreuzes in der Form R,G,B
CrossPadding	Checkboxen	Abstand des Kreuzes vom Checkbox-Rand in Mikrometern
CrossWeight	Checkboxen	Strickdicke des Kreuzes in Mikrometern
DataField	Eingabeobjekte, Bilder	Name des assoziierten Datenfeldes der Datenquelle beim Datenbankzugriff

Eigenschaft	Objekttypen	Bedeutung
DataType	Eingabefelder	Datentyp des Eingabefeldes: 1 Text (DATATYPE_TEXT) 2 Ganzzahl (DATATYPE_INT) 4 Gleitkommazahl (DATATYPE_FLOAT) 8 Datum (DATATYPE_DATE) 16 Uhrzeit (DATATYPE_TIME) Die Werte sind in *formdll.h* als Konstanten definiert
Default	Eingabeobjekte	Vorbesetzung als Text (bei Checkboxen 0 oder 1)
Descriptor	Tabellen	Deskriptor-String, der die Struktur der Tabelle definiert. Tabellen-Deskriptoren sind an gesonderter Stelle beschrieben.
DsName	Odbc-Zugriff	Name der Datenquelle
DsPassword	Odbc-Zugriff	Passwort der Datenquelle
DsUserId	Odbc-Zugriff	Benutzerkennung
EnableFlags	Odbc-Zugriff	Bitliste der Operationen, für die ein Button beim Datenzugriff angeboten wird: 1 Matchmode ein/aus 2 Auf Feldanfang selektieren 4 Auf exakten Feldinhalt selektieren 8 Standardmaske zum Selektieren 16 Alle Datensätze selektieren 32 Selektion aufheben 64 Selektion aufheben 256 Sortieren 4096 Navigation (4 Buttons) 65536 Einfügen 131072 Löschen 262144 Zurückschreiben
Format	Alle	Format-Deskriptor, der die Datenformatierung definiert. Format-Deskriptoren sind in einer gesonderten Dokumentation beschrieben.

Eigenschaft	Objekttypen	Bedeutung
FrameSize	Alle	Rahmendicke in Mikrometern. Nur relevant, wenn *FrameType* nicht 0 ist.
FrameType	Alle	Art des Rahmens: 0 kein Rand 1 durchgehende Linie 2 gepunktete Linie 3 3D-Rand
GroupId	Alle	ID die alle Objekte derselben Gruppe assoziiert.
HelpText	Buttons	Hilfetext, der angezeigt wird, wenn der Button mit der Action 5 definiert ist. (Eigenschaft *Action)*
Hint	Eingabeobjekte	Text, der in der Statuszeile angezeigt wird
HtmlLink	Alle	Mit dem Objekt assoziierter Link auf eine URL. Nur relevant für den Export nach HTML.
JumpNext	Eingabeobjekte	Definition des Sprungverhaltens, wenn in einem Eingabeobjekt die Tabulatortaste betätigt wird: 1 auf das nächste Eingabeobjekt springen 2 auf das explizit angegebene Objekt springen 3 piepen und nicht springen 4 nicht springen 5 das Sprungziel wird durch ein Makro ermittelt
JumpPrev	Eingabeobjekte	Definition des Sprungverhaltens, wenn in einem Eingabeobjekt die Rückwärts-Tabulatortaste betätigt wird: 1 auf das vorige Eingabeobjekt springen 2 auf das explizit angegebene Objekt springen 3 piepen und nicht springen 4 nicht springen 5 das Sprungziel wird durch ein Makro ermittelt
LabDist	Checkboxen	Abstand zwischen Label und Checkbox in Mikrometern

Eigenschaft	Objekttypen	Bedeutung
Label	Checkboxen	Label, das links oder rechts neben der Checkbox angezeigt wird.
Mask	Bilder	Maskentyp, mit der das angezeigte Bild maskiert wird: 1 keine Maske 2 Ellipse 3 Kreis
Meaning	Eingabeobjekte	Semantische Bedeutung des Eingabeobjekts. Auf Basis dieser Bedeutung werden Exporte durchgeführt (z.B. als Visitenkarte): 0 keine spezifische Bedeutung 1 Nachname 2 Vorname 3 Mailadresse 4 Private Telefonnummer 5 Dienstliche Telefonnummer 6 Mobiltelefonnummer
Name	Alle	Name des Objekts. Dieser wird insbesondere benötig, um das Objekt per Makro anzusprechen.
NextObjName	Eingabeobjekte	Name des Eingabeobjekts, das angesprungen wird, wenn die Tabulator gedrückt wird und wenn die Eigenschaft *JumpNext* den Wert 2 (explizites Objekt anspringen) hat.
NonCheckText	Checkbox	Wert, der mit der Checkbox im angekreuzten Zustand assoziiert ist als Text
OrderBy	Odbc-Zugriff	ORDER BY Klausel
Outline	Alle	Position und Größe des Objekts als Rechtck in der Form: L,T,R,B (left, top, right, bottom)
Padding	Alle	Innerer Randabstand in Mikrometern
PictureFile	Bilder	Name der Bilddatei, falls das Bild den Quelltyp 2 hat (siehe Eigenschaft *Source)*

Eigenschaft	Objekttypen	Bedeutung
PrevObjName		Name des Eingabeobjekts, das angesprungen wird, wenn die Rücktabulatortaste gedrückt wird und wenn die Eigenschaft *JumpPrev* den Wert 2 (explizites Objekt anspringen) hat.
RadioGroupName	Checkboxen	Name der Gruppe, die alle sich gegenseitig auslösenenden Checkboxen zusammenfasst
Rop	Bilder, Buttons	Rasteroperation, mit der das Bild angezeigt wird: 1 And 2 Copy
Select	Odbc-Zugriff	SELECT Statement
Semantic	Eingabefeld	Semantik-Descriptor, der die semantische Prüfung des Eingabefeldes kontrolliert. Der Deskriptor-String ist in einer gesonderten Dokumentation erkläutert.
ShadowSize	Rechtecke	Dicke des 3D-Schattens in Mikrometern
Shape	Checkboxen	Form der Checkbox: 0 Rechteck 1 Kreis
SizeMode	Bilder	Art der Einpassung des Bildes in die Äußere Form; 1 Bild einpassen, Seitenverhältnis variabel 2 Bild einpassen, Seitenverhältnis beibehalten 3 Bild in Originalgröße anzeigen 4 Bild mit einem Vergrößerungsfaktor anzeigen
Source	Bilder	Typ der Bildquelle: 0 Kein Bild 1 Eingebettetes Bild 2 Referenz auf Bilddatei 3 Bildzugriff aus Datenquelle

Eigenschaft	Objekttypen	Bedeutung
Special	Rechtecke	Bitliste, die Pfeilspitzen für das Rechteck definiert, in hexadezimaler Form: 0x00000000 Die Bits haben folgende Bedeutung: 0x00001 links 0x00002 oben 0x00004 rechts 0x00008 unten
State	Alle	Bitliste, die Optionen für Objekte spezifiziert, in hexadezimaler Form: 0x00000000 Die Bits haben folgende Bedeutung: 0x00000001 Objekt drucken 0x00000002 Objekt anzeigen 0x00000004 Position nicht veränderbar 0x00000008 Größe nicht veränderbar 0x00000010 STATE_ACTIVE 0x00000020 Gedrückt (nur für Buttons) 0x00000040 Quadratische Form erzwungen 0x00000080 Festes Seitenverhältnis 0x00000100 Bezugsposition: Quadrant 1 bis 3 0x00000200 Hintergrund transparent 0x00000400 Es muss eine Auswahl erfolgen 0x00000800 Objektinhalt ist ein Dateiname 0x00001000 Automatisch vertikal rollen 0x00004000 Intern für GUI-Refresh 0x00008000 Automatisch horiziontal rollen 0x00010000 Mehrzeilig 0x00020000 ReadOnly, kann nicht betreten werden 0x00040000 Großschreibung 0x00080000 Rand oben offen 0x00100000 Keine Unterteilung 0x00200000 Beim Datenzugriff zurückschreiben 0x00400000 ReadOnly, darf aber betreten werden 0x00800000 Automatisch eine Liste erzeugen 0x01000000 Sortieren 0x02000000 Automatisch Daten Selektieren

Eigenschaft	Objekttypen	Bedeutung
		0x04000000 Muss gefüllt sein 0x08000000 Automatisch aufklappen
Style	Rechtecke	Stil des Rechtecks: 1 flach 2 3D erhaben 3 3D vertieft 4 transparent
TempDispColor	Odbc-Zugriff	Farbe des Fensters, in dem die Operationsanzeige erscheint in der Form R,G,B
TempDispTime	Odbc-Zugriff	Dauer der Einblendung der Operationsanzeige in 1/17 Sekunden
Text	Alle	Anzeigetext (Ausgabeobjekte), bzw. Inhalt (Eingabe-objekte)
Type		Typ des Objekts als ganze Zahl: 1 Textobjekt (FOTYPE_TEXT) 2 Rechteck (FOTYPE_RECT) 4 Linie (FOTYPE_LINE) 8 Kreis (Ellipse) (FOTYPE_CIRCLE) 16 Bild (FOTYPE_PICTURE) 32 Eingabefeld (FOTYPE_EDIT) 64 Ankreuzfeld (FOTYPE_CHECKBOX) 128 Auswahlfeld (FOTYPE_COMBO) 256 Datum (FOTYPE_DATE) 512 Uhrzeit (FOTYPE_TIME) 1024 Ausfertigung (FOTYPE_COPYNAME) 2048 RTF-Objekt (FOTYPE_RTF) 4096 Button (FOTYPE_BUTTON) 32768 Odbc-Zugriff (FOTYPE_ODBCACC) 65536 Datei-Zugriff (FOTYPE_FILEACC) 131072 Tabelle (FOTYPE_TABLE) Die Objekttypen sind Bits. Die Werte sind in *formdll.h* als Konstanten definiert.

Eigenschaft	Objekttypen	Bedeutung
TypeName		Sprechender Name, der den Typ des Obkekts benennt, z.B. *Eingabefeld* oder *Auswahlfeld*
Where	Odbc-Zugriff	WHERE Klausel
Zoom	Bilder	Vergrößerungsfaktor, mit dem das Bild dargestellt wird in Prozent.