2 1.1. Vorwort

Mit der SAA-Toolbox stehen Ihnen C-Funktionen zur Verfügung, die Ihnen helfen werden, Ihre Programme mit einer professionellen Benutzerschnitstelle auszustatten, ohne selbst »das Rad noch einmal erfinden zu müssen«. Ausgetestete und auf Geschwindigkeit und leichte Bedienbarkeit hin optimierte Routinen geben Ihrer Software das »look and feel« der neuen und die Zukunft bestimmenden Software-Generation, die sich an dem SAA-Standard orientiert:
+ Leichte Bedienung des Anwenderprogramms durch Verwendung einer Menüzeile und Pull-down-Menüs,
+ wahlweise Bedienung des Anwenderprogramms mit Maus und Tastatur,
+ das Arbeiten mit mehreren Fenstern,
+ Programmsteuerung mit Hilfe von Dialogfelder
sind nur einige der Merkmale dieses neuen Standards, den Sie vielleicht von der Shell des Betriebssystems DOS 4.0 oder der Grafik-Version von Microsoft Windows 2.1 oder /386 her kennen. Natürlich folgt auch QuickC diesem Standard, und im Laufe dieser Dokumentation werden Sie alte Bekannte aus der QuickC-Bedieneroberfläche wiederfinden.
Neben der umfangreichen Funktionssammlung werden die Elemente des SAA_Standards erläutert, die eine Vereinheitlichung der Benützeroberfläche zum Ziel haben. Weiter Informationen zur System Anwendungs-Architektur können Sie den Veröffentlichungen von IBM zu SAA entnehmen (siehe Literaturverzeichnis). Allein die SAA-Toolbox besteht bereits aus mehr als 30.000 Zeilen ausführlich dokumentierten Source-Codes.
Die Ihnen vorliegende SAA-Toolbox-Dokumentation wird es Ihnen ermöglichen, daß Sie auf all diese Funktionen über die Ihnen zur Verfügung stehenden Bibliotheken zugreifen können.
In diesem Kapitel erhalten Sie einen ersten Überblick über die Leistungsmerkmale der Toolbox. Außerdem werden wir die im Text und in den Programmen verwendeten Notationsregeln erläutern und Sie mit dem Aufbau der Dokumentation vertraut machen.

1.2. Die SAA-Toolbox im Überblick

Die Tools unterstützen Sie bei der Programmentwicklung einer anwenderfreundlichen Benutzerschnittstelle, die sich an dem von IBM definierten SAA-Standard (System-Anwendungs-Architektur) anlehnt. Durch die Verwendung der Toolbox wird die Dauer für die Erstellung einer Bedieneroberfläche, die den Großteil der Programmierzeit beträgt, erheblich verkürzt. Zur Verfügung stehen Ihnen:
SAA-Toolbox:
+ Ein proffesioneller Menümanager, der vom Anwender mit Maus und Tastatur bedient werden kann.
+ Ein komfortables Window-Management für die Generierung, Anzeige und Verwaltung von mehreren Fenstern, die sich beliebig überlappen können.
+ Eine Sammlung von Dialogfeldern, die Sie über einen simplen Funktionsaufruf in Ihre Programme einbinden können.
+ Eine Funktionssammlung zur Unterstützung der Microsoft Maus.
+ Schnelle Assembler- und C-Routinen zur Bildschirmausgabe, die direkt auf den Bildschirmspeicher zugreifen.
MASK-Toolbox:
+ Ein Maskencompiler und C-Funktionen, die die vom Maskencompiler erstellten Maskendateien interpretieren
+ Ein Formularcompiler und C-Funktionen, die die erstellten Formulare und Listen für den Ausdruck interpretieren.
ISAM-Toolbox:

+ Die Funktionen, die Ihnen die Datenbankverwaltung zur Verfügung stellt, verwenden als Basis das Novell-Btrieve-Tool.
UTIL-Toolbox:
+ Weiters stehen Ihnen Funktionen zur Stringbearbeitung und Low-Level-Funktionen aller Art zur Verfügung.
Bei der Realisierung der Funktionen haben wir versucht, die bestmögliche Kombination von Effektivität und leichter Bedienbarkeit zu finden.
Die verschiedenen Funktionen der SAA-Toolbox sind in folgenden Kategorien zusammengefaßt:

Präfix	Kategorie	Header-Dateien
	Lädt alle anderen Header-Dateien	eur_tool h
Dl_	Funktionen zur Aktivierung der Dialogfelder	eur_dlg h
Mn_	Funktionen zur Initialisierung und Aktivierung des Menümanagers	eur_mnu h
Ms_	Funktionen zur Unterstützung der Microsoft-Maus	eur_msm h
Ut_	Utility-Funktionen und Makros	eur_utl h
Vi_	Funktionen zur schnellen Bildschirmausgabe	eur_vio h
Wi_	Funktionen zum Window-Handling	eur_win h
i_	interne SAA-Funktionen	eur_int h
	Funktionstypen, typedef,	eur_type h
	Funktionen für Stringmanipulationen	eur_str h
	DOS Low-Level Funktionen	eur_dos h
BTRV B_	Btrieve ISAM-Funktionen	eur_btr h
M_	Masken-Interpreter-Funktionen	eur_tool h

Tabelle 1.1: Die Toolbox-Kategorien und die Präfixe der Funktionsnamen.
Die Spalte »Präfixe« der Tabelle 1.1. informiert Sie über die Vorsilben, die den einzelnen Funktions- und Makronamen vorangestellt sind, um sie den Kategorien zuzuordnen.
So enthält das Modul zur Steuerung des Menümanagers (Präfix: Mn_) eine Funktion, mit der Menüs initialisiert werden können: »Mn_Einrichten()«.
Außerdem informiert Sie dieses Präfix darüber, welche Deklarationsdatei von Ihnen zur Typenprüfung der Funktionsargumente in Ihren Code miteingebunden werden muß.
Wenn Sie beispielsweise in einem Programmteil die Toolbox-funktion »Mn_Einrichten()« verwenden wollen, müßen Sie die Header-Datei »eur_mnu.h« miteinbinden.
Wenn Ihr Programm mehrere Header-Dateien der Toolbox benötigt, können Sie auch

die Header-Datei »eur_tool« verwenden. Diese Datei bindet sämtliche Header-Dateien der Toolbox mit einer anweisung ein. Durch die Präprozessorbefehle zur bedingten Kompilation ist sichergestellt, daß jede Header-Datei nur einmal eingebunden wird.
Bei der Namensvergabe der Funktionen haben wir uns die Fähigkeit des Microsoft-C Compilers und des Linkers zunutze gemacht, 31 siginifikante Stellen zu unterscheiden. Die Funktionsnamen wurden dadurch zwar länger, verdeutlichen jedoch gleichzeitig besser den Verwendungszweck der Funktion. (Beispiel: Ms_CursorOff(): schaltet den Maus-Cursor aus).
Die Funktionsnamen verwenden eine Mischung aus Klein- und Großschreibung, wobei diese Notation die Funktionsnamen in einzelne Worte teilt.
Für Low-Level Funktionen, DOS-Funktionen oder Funktionen zur Stringmanipulation verwenden wir, gleich wie die Microsoft-Standard-Biblitohek, nur Kleinbuchstaben.
Interne SAA-Toolbox-Funktionen beginnen mit »i_«. Die Funktionen des Maskeninterpreters verwenden ein Präfix »M_« und die Btrieve-Funktionen das Präfix »B_«.

1.3. Notationsregeln

Zur Angabe des Datentyps einer Variable verwenden wir Datentypen, die mit der »typedef-Anweisung« geschaffen wurden. Ziel dabei ist es, Datentypen, die in C nicht explizit vorhanden sind, zu schaffen (z.B. den Boolschen Datentyp) als auch durch die Vergabe von Großbuchstaben den Code übersichtlicher zu gestalten.
Die wichtigsten Datentypen finden Sie in der nachfolgenden Tabelle. Eine komplette Übersicht können Sie der Header-Datei »eur_type.h« entnehmen. Wenn Sie bereits unter Windows programmiert haben, werden Ihnen diese »Typen« bekannt vorkommen.

Toobox-Typ	C-Typ	Verwendungszweck
CHAR	unsigned char	Zeichen
BYTE	unsigned char	Zahl im Bereich 0-255
WORD	unsigned int	Zahl im Bereich 0-65.535
LONG	unsigned long	Zahl im Bereich 0-4.294.967.295
BOOL	signed short int	Boolsche Variable:TRUE/FALSE
PSTR	unsigned char *	Zeiger auf String
FPSTR	unsigned char far *	far-Zeiger auf String
PWORD	unsigned int *	Zeiger auf Word
FPWORD	unsigned word far *	far-Zeiger auf Word

Tabelle 1.2: Die wichtigsten Toolbox-Datentypen

Den Toolbox-Datentypen für vorzeichenbehaftete C-Typen wird ein »S« (für »signed«) vorangestellt. Somit erhält beispielsweise eine Variable vom C-Typ »signed int« den Toolbox-Typ SWORD.
Ein große Manko vieler Programme sind Variablennamen ohne jede Aussagekraft. Ihre Programmierer verwenden offensichtlich keine oder mur wenig Zeit für die Namensgebung der Variablen. Dies führt dann in der Regel dazu, daß die so entstandenen Programme für außenstehende Personen nicht mehr zu durchschauen sind und daher kaum gewartet werden können.
Um die Variablennamen der Toolbox aussagekräftiger zu machen, verwenden wir eine Nomenklatur, die sich an die »ungarische Namenskonvention« anlehnt, die vom Microsoft-Programmierer Charles Simony entwickelt worden ist. Mit Ihr soll erreicht werden, daß aus einem Variablennamen sowohl die Aufgabe der Variable

im Programm als auch Ihr Datentyp hervorgeht.
Der Datentyp wird dabei durch ein Präfix vor dem Variablennamen angegeben., das den Typ der Variable eindeutig beschreibt. Dieses Verfahren wirkt auf den ersten Blick sicherlich etwas abschreckend, es hat sich jedoch gezeigt, daß nach einer kurzen Gewöhnungsphase die Vorteile bei weitem überwiegen.
In der folgenden Tabelle finden Sie eine Aufstellung der wichtigsten Präfixe. Eine vollständige Übersicht ist in der Datei »eur_type.h« enthalten.

Präfix	Datentyp	Bedeutung
c	CHAR (unsigned char)	Zeichen
b	BYTE (unsigned char)	Zahl im Bereich 0-255
w	WORD (unsigned int)	Zahl im Bereich 0-65.535
bo	BOOL (unsigned short int)	Boolsche Variable:TRUE/FALSE
f	FLOAT (float)	Fließkommazahl
d	DOUBLE (double)	Fließkommazahl dopp. Genauigk.
pstr	PSTR (unsigned char *)	Zeiger auf String
fpstr	FPSTR (unsigned char far *)	far-Zeiger auf String
pw	PWORD (unsigned int *)	Zeiger auf Word
fpw	FPWORD (unsigned int far *)	far-Zeiger auf Word

Tabelle 1.3: Präfixe zur Typenangabe in Variablennamen
Besonder Behandlung erfahren dabei Registervariablen und andere Shleifenzähler sowie die programmglobalen und modulglobalen Variablen und Array.
Registervariablen, die meist zur Optimierung von Schleifen verwendet werden, erhalten Variablennamen, die aus einem Buchstaben bestehn. Das gleiche gilt für Zählvariablen, die immer vom Typ WORD (unsigned int) sind. Für diese Variablen werden immer die Buchstaben »i, j, k...« verwendet.
Allen programmglobalen Variablen wird zusätzlich die Nachsilbe »_g« angehängt. Wenn Sie im Source-Code beispielweise eine Variable mit dem Namen »wAktMnu_g« entdecken, handelt es sich um eine globale Variable vom Typ WORD. Modulglobale Variablen, die außerhalb der Funktion dieser Datei definiert werden, und somit nur beim Programmstart initialisiert weren, erhlaten als Namensergenzung die Kennzeichnung »_m«. Eine Variable mit dem Namen »wInit_m« ist somit modulglobal und hat den Datentyp WORD.
Arrays werden durch Ergänzung des Präfixes um den Buchstaben »a« gekennzeichnet. Der Variablennamen acRahmenzeichen_g kennzeichnet also ein globales Array aus Zeichen (CHAR).
Neben den Funktionen, die wir Ihnen als Programmierer zur Verfügung stellen, verwendet die Toolbox interne Funktionen, die Sie am Präfix »i_« (für intern) erkennen können. Die Wirkungsweise der Funktionen wird in der Funktionsdokumentation ausführlich beschrieben. Da diese Funktionen die Datenstruktur der Toolbox manipulieren und sehr nah an die aufrufenden Funktionen angelehnt sind, empfehlen wir Ihnen, diese Funktionen nicht zu benutzen. Die Prototypen dieser Funktionen finden Sie in der Datei »eur_int.h«. Da diese Datei nur intern verwendet wird, brauchen Sie diese Deklarationen nicht in Ihr Programm einzubinden.
Eine Tabelle aller Funktionsnamen finden Sie im Anhang A dieser Dokumentation.

1.4. Die Bibliothek

Sollten Sie mit der Programmiersprache C und Ihrem Microsoft C-Compiler noch nicht so vertraut sein, empfehlen wir Ihnen das Buch »Programmieren mit QuickC«, erschienen in der »Microsoft Edition« im Verlag Markt&Technikik.
Die Einzelbibliotheken der SAA-Toolbox sind für den Einsatz der verschiedenen Speichermodelle vorbereitet. Alle Bibliotheken beginnen mit dem Präfix »saa_« (für SAA-Toolbox). Die Dateien werden durch ein Suffix aus vier Zeichen unterschieden. Die ersten drei Zeichen geben den Compiler un desen

Versionsnummer an, mit dem die Bibliotheken erstellt wurden:
M51 = Microsoft C-Compiler, Version 5.1
Das letzte Zeichen dieses Suffixes liefert Informationen über das Speichermodell, mit dem die Bibliothek eingesetzt werden kann:

Symbol	Speichermodell	Daten	Code
S	Small	64 Kbyte	64 Kbyte
M	Medium	64 Kbyte	1 Mbyte
C	Compact	1 Mbyte	64 Kbyte
L	Large	1 Mbyte	1 Mbyte

Tabelle 1.4: Bedeutung des Suffixes im Bibliotheksnamen

So bedeutet beispielsweise der Dateiname »saa_m51l.lib« daß es sich um die Bibliothek der SAA-Toolbox von euroSOFT handelt, die mit dem Microsoft C-Compiler 5.1 für das Large-Speichermodell erstellt wurde.
Diese Namenskonfention trifft auch auf die anderen Bibliotheken im Tool zu. Zum Beispiel enthält die Bibliothek »eur_m51l.lib« (für euroSOFT-Toolbox) die Funktionen für die ISAM-Routinen und die Stringmanipulationen. Die Bibliothek »dos_m51l.lib« enthält Low-Level Dos-Funktionen.
Wegen der Kompatibilität des Microsoft C-Compilers und des QuickC-Compilers können alle Bibliotheken sowohl mit QuickC als auch Microsoft C5.1 verwendet werden. Beim QuickC-Compiler ergeben sich im Dialogmodus jedoch sehr schnell Speicherplatzprobleme.

1.5. Die Installation

Wir empfehlen Ihnen folgende Verzeichnisse anzulegen:

Directory	Inhalt
\euro\t01	
\exe	
\dat	
\hlp	
\hdb	
\loc	
\lib	
\include	
\bin	
\btrieve	

1.1. Vorwort
2.1. Vorwort
IBM hat mit der Vereinheitlichung der Benutzeroberflächen, Programmierschnittstellen und der Kommunikationsprotukolle für die drei IMB Hardware-Architekturen PC, /3X und /370 wegweisende Schritte für die Zukunft der Software auch auf dem Personalcomputer gesetzt.
Die drei Komponenten werden unter dem Stichwort SAA (System Anwendungs-Architektur) zusammengefaßt.
Wir wollen Ihnen die Bedeutung und die Grundlagen des SAA-Konzeptes vorstellen. Unser Hauptaugenmerk bei der Dokumentation zur SAA-Toolbox richten wir natürlich auf die SAA-Komponente »einheitliche Benützerführung«.
Sie werden bei der ISAM-Toolbox sehen, daß wir auch für den Dateiaufbau einen Standard als Basis verwenden. Als Datenbankgrundlage dient der Standard von Novell: Btrieve, doch dazu später.
2.2 Die einheitliche Anwendungsunterstützung
Für Programmierer und Softwarehäuser ist die einheitliche Anwendungsunterstützung nach SAA nicht nur aus der Sicht des Kunden wichtig. IBM plant, die Werkzeuge für die Programmentwicklung auf den verschiedenen IBM Hardware-Architekturen zu vereinheitlichen. Dadurch wird eine Portabilität der entwickelten Software auch zwischen den Hardware-Systemen erreicht werden. Programme, die auf einem der Systeme entwickelt wurden, sollen auch auf allen anderen Architekturen eingesetzt werden können. Dadurch wird der Entwicklungsaufwand für die Software-Häuser erheblich reduziert.
Gleichzeitig kann sich der Programmentwickler voll auf die zu erstellende Anwendung konzentrieren, ohne auf die Eigenheiten des speziellen Betriebssystems un der konkreten Hardware Rücksicht nehmen zu müssen.
Ein Beispiel für diese Programmierschnittstelle ist das API (application program interface) für den Presentation Manager des Betriebssystems OS/2. Anwendungen, die mit diesem Werkzeug erstellt werden, sollen in Zukunft ohne große Schwirigkeiten auf alle IBM Hardware-Architekturen portierbar sein.