

FIX/Win

Release 3.1.0

Handbuch für Anwendungsentwickler

Vorwort

Dieses Handbuch beschreibt FIX/Win in der Version 3.1.0.

FIX/Win wird entwickelt und vertrieben von

Nonne & Schneider Informationssysteme GmbH
Friedrich List Straße 31
35398 Gießen
Tel: 0641/97477-0, Fax:0641/97477-77

Im Zuge der Weiterentwicklung von FIX/Win können Leistungsmerkmale hinzukommen, verändert werden oder entfallen.

Jegliche Vervielfältigung dieses Buches, Übersetzungen, Nachdrucke u.dgl. auszugsweise und auch gesamt, sind nur mit ausdrücklicher Genehmigung des Herstellers gestattet.

Die Nichterwähnung von Warenzeichen, Gebrauchsmustern etc. berechtigt nicht zu der Annahme, eine Ware, ein Name etc. sei frei.

Handbuchversion: Edition 3.1.0b Nonne & Schneider Informationssysteme GmbH, 1997, 1998, 2002, 2007, 2008

1	Allgemeiner Überblick	9
1	Was ist FIX/Win?	9
2	Voraussetzungen für den Einsatz	9
3	Konzepte von FIX/Win	9
3.1	Programmablauf von FIX/Win	9
3.2	Elemente von FIX/Win	11
3.3	Versionsangabe von FIX/Win	12
3.3.1	Ermitteln der Versionsangabe	13
3.4	Anpassen und Erweitern von FIX/Win	14
4	Installation	16
5	Verzeichnisaufbau und Komponenten von FIX/Win	17
6	Grundlegende Konfiguration	19
6.1	Kommandozeilenparameter	19
6.2	Globale Einstellungen von FIX/Win	21
6.3	Einrichten von FIX-Anwendungen	23
6.3.1	Eintragen des Programmes	23
6.3.2	Bearbeiten der Programmtabelle mit "edprg"	23
6.3.3	Erstellen eines Startskripts	24
2	Arbeiten mit FIX/Win	27
1	Bedienung mit der Tastatur	28
2	Arbeiten mit der Maus	29
2.1	Die IconBox	29
2.2	Mauszeiger unter FIX/Win	29
3	Aufbau und Funktion des Hauptmenüs	31
3.1	Hauptmenüpunkt "FIX/Win"	31
3.2	Hauptmenüpunkt "Anzeige"	32
3.3	Hauptmenüpunkt "Einstellungen"	36
3.4	Hauptmenüpunkt "Hilfe"	38
4	Ausgabe von Listen	39
3	Anpassung von FIX/Win	41
1	Kodierung der Konfigurationsdateien	41
2	Aufbau eines Gruppenverzeichnisses	41
2.1	Verfahren zum Einlesen von Dateien	44
3	Konfiguration der IconBox	45
3.1	Der grundsätzliche Aufbau der IconBox	45
3.1.1	Aufbau der Icondefinitionsdatei	46
3.1.2	Vorgehensweisen bei Änderungen	48
3.2	Aussehen der IconBox	49
3.2.1	Einstellungen über Ressourcen	50
3.2.2	Bitmaps für Icons	50
3.2.3	Bitmaps für Rahmen	51
3.3	Umschalten des Mauscursor bei maussensitiven Masken	51
4	Erstellen einer eigenen Tastenbelegung	53
4.1	Tasten-Label	54
4.2	Rahmen für Tasten-Labels	56
4.3	Beispiel einer neuen Tastenbelegung	56
5	Verwendung von Semigrafikzeichen	57
5.1	Beispiele zur Erstellung eigener Semigrafikzeichen	58
6	Packen von Bitmaps	60
7	Erstellen eigener Farbzusordnungstabellen	63
8	Anpassung von Schriftarten	67
8.1	Verwendung von Schriftarten des Windowssystems	67

8.1.1	Informationsdateien für Fullwidth-Zeichen	68
8.2	Verwendung der mitgelieferten Schriftarten	71
8.3	Definition weiterer Schriftarten	72
9	Anpassung der Schreibmarke	74
10	Anpassen von Meldungen, Menüpunkten und Dialogen	74
11	Bedienung durch das Entwicklermenü	75
11.1	Menüpunkte des Entwicklermenüs	75
12	Möglichkeiten des Diagnosemenüs	77
13	Verwendung von FIX/Win-Ressourcen	79
13.1	Allgemeines zu Ressourcen	79
13.2	Ressourcendateien von FIX/Win	80
13.3	In FIX/Win verwendete Ressourcen	82
13.4	Ausgabe von Tönen	84
13.5	Steuerung des Busy-Cursors	85
13.6	Vorgabe von Ports	85
13.7	Definition eines Signals für den Verbindungsabbruch	85
13.8	Anzeige eines Logos und Abspielen eines Tons	85
13.9	Timeout bei der Fehlerausgabe	86
13.10	Zeichensatz von stderr und stdout	86
13.11	Konfiguration des Größen-/Stilmenüs	86
13.12	Konfiguration von Tooltips	87
13.13	Starten eines Editors	87
4	Erstellen einer Benutzerbibliothek	89
1	Allgemeine Hinweise	89
1.1	Laden der Benutzerbibliothek	89
1.2	Include-Dateien	89
1.3	Arten von Funktionen	89
1.4	Parameter CallBack	90
1.5	Parameter p_callID	90
2	Beschreibung der FWown-Funktionen	91
3	Beschreibung der FIX/Win-Funktionen	98
3.1	Funktionen des FIX/Win Kerns	98
3.2	Funktionen des FIX/Win Standard-Frameworks	101
5	Spezielle Features von FIX/Win	105
1	Anzeige von modalen Dialogen in der Benutzerbibliothek	105
2	Verstecken des Hauptfensters von FIX/Win	106
3	Ausgaben des Meldungsfensters umleiten	106
3.1	Abschalten des Meldungsfensters	106
3.2	Abfangen von Meldungen	106
3.3	Beispiel	107
4	Erstellen von Protokoll-Profilen	108
4.1	Funktionsweise des Protokolls	108
4.2	Einschalten von Protokoll-Profilen	108
4.3	Aufbau der Datei	109
4.4	Optimierungen	112
4.4.1	Refresh	112
4.4.2	Paintarea	114
4.4.3	Fullwidth-Zeichen	114
4.4.4	Weitere Optimierungen	115
6	Fehlermeldungen	117
1	BM - Bitmaps	117

2	CN - Verbindungsaufbau	118
3	DV - Developer	119
4	FX - FIX bzw. FIX/Win allgemein	119
5	IB - IconBox	120
6	IM - Iconman	122
7	KL - Keylabels (Tasten-Label)	125
8	LG - Logo	127
9	NT - Netzwerk	127
10	OB - Objekte	129
11	PR - Protokoll	130
12	PT - Drucken	131
13	RS - Ressourcen	131
14	VC - Virtueller Text-Bildschirm	132
15	VG - Virtueller Grafik-Bildschirm	132
7	Migrationshinweise	135
1	Allgemeines	135
2	Zeichensatz der Konfigurationsdateien	135
3	Ressourcedatei mit globalen Einstellungen	136
4	Zeichensatz von stderr und stdout	136
5	Länderspezifische Einstellungen	136
6	Verwenden von Schriftarten des Windowssystems	136
7	Format der gepackten Bitmaps	137
7.1	Neue Bitmaps	137
8	Neue Einträge für Felder der Farbuordnungstabelle	137
9	Belegung der Taste ESC	137
10	Änderungen an Windows-Ressourcen	137
11	Anpassen der Benutzerbibliothek	138
12	Erstellen von Protokoll-Profilen	139
13	Starten eines Editors	140
Anhang H	Index	141

1 Allgemeiner Überblick

1 Was ist FIX/Win?

FIX/Win ist ein grafisches Frontend für Microsoft Windows zur Bedienung von FIX-Anwendungen. Die FIX-Anwendung benötigt dabei keine besonderen Anpassungen, außer dass sie mit FIX ab Release 3.1.0 kompiliert wurde. So ist es möglich, dass dieselbe Anwendung sowohl unter einer grafischen Oberfläche als auch auf einem Textbildschirm abläuft. Allerdings bietet FIX/Win über den Terminalbetrieb hinausgehend folgende Features:

- ansprechender grafischer Aufbau des Bildschirms
- Verwendung der Maus zur Bedienung von FIX-Elementen
- Darstellung des FIX-Bildschirms in verschiedenen Größen
- Darstellung des FIX-Bildschirms in verschiedenen Farbstilen
- Darstellungen von Teilen des FIX-Bildschirms durch eigene Zeichenfunktionen, Verwendung von proportionalen Schriften.
- Darstellung von Symbolen und Logos
- Bedienung über anklickbare Icons
- Definition von Icons, eigenen Tastenbelegungen, Farbstilen, Symbolen und Logos durch den FIX-Anwendungsentwickler
- Einbinden eigener Funktionen und somit auch die Möglichkeit, Programme auf dem Rechner zu starten, auf dem FIX/Win gestartet wurde.

2 Voraussetzungen für den Einsatz

Folgende Voraussetzungen müssen für FIX/Win gegeben sein:

- Betriebssystem: Windows 2000, Windows XP oder Windows 2003
- freier Festplattenspeicher: ca. 16 MB für die komplette Installation oder ca. 3 MB für das Runtime-Paket
- ca. 4- 5 MB freier Hauptspeicher
- FIX-Anwendung, die mit FIX Release ab 3.1.0 erstellt wurde

3 Konzepte von FIX/Win

3.1 Programmablauf von FIX/Win

FIX/Win ist ein um grafische Elemente angereichertes Frontend zur Bedienung von FIX-Anwendungen unter Windows. Es sind bis zu drei Rechner am Lauf von FIX/Win und FIX Anwendung beteiligt. Die Aufgaben dieser Rechner können jedoch auch von einem einzigen Rechner übernommen werden.

- Applikationsserver Auf diesem Rechner wird die FIX Anwendung gestartet.
- Frontend-Host Der Rechner auf dem FIX/Win gestartet wird.
- Display-Host Der Rechner, auf dem die Anzeige von FIX/Win dargestellt wird. Wenn mit einem Terminal-Server gearbeitet wird, handelt es sich dabei um einen Client-Rechner, der FIX/Win über RemoteDesktop bedient. Ansonsten handelt es sich bei Display-Host und Frontend-Host um ein und denselben Rechner.

Die FIX-Anwendung wird auf dem Applikationsserver gestartet, während FIX/Win auf dem Frontend-Host für die grafische Aufbereitung des Bildschirms und das Erkennen von Eingaben verantwortlich ist. Damit dies funktioniert, ist es notwendig, eine Netzwerkverbindung zwischen Frontend-Host und Applikationsserver aufzubauen. FIX/Win wendet sich dazu an den Service "rexec"¹ auf dem Applikationsserver, der nach dem Zustandekommen der Verbindung die FIX-Anwendung startet. Die FIX-Anwendung sendet dann über die eingerichtete Netzverbindung nach einem fest vorgegebenen Protokoll ihre Bildschirmausgaben. Diese werden von FIX/Win umgesetzt und grafisch aufbereitet ausgegeben. Bezüglich der Ausgabe sind in FIX/Win zwei grundsätzliche Einstellungen möglich (siehe auch [Hauptmenüpunkt "Einstellungen" auf Seite 36](#)):

- *Bildgröße* FIX-Programme benötigen in der Regel 25 Zeilen und 80 Spalten. Die Abmessungen des Ausgabefensters von FIX/Win hängen von diesen Werten und der Schriftgröße ab. In FIX/Win lassen sich drei verschiedene Schriftgrößen und somit drei Bildgrößen einstellen. Sie werden in dieser Dokumentation *Medium*, *Large* und *Huge* genannt.
- *Bildstil* Bestimmte FIX-Elemente werden von FIX/Win in verschiedenen Farben dargestellt. Diese Farben sind in einem *Bildstil* zusammengefasst. Zusätzlich enthält ein *Bildstil* einen Satz von Bitmaps zur Darstellung von Semigrafikzeichen. In FIX/Win lassen sich vier verschiedene Bildstile einstellen. Jeder dieser Stile kann vom Anwendungsentwickler in Bezug auf das optische Erscheinungsbild neu entworfen oder verändert werden.

FIX/Win setzt alle Eingaben in FIX-Events um und sendet sie über die Netzwerkverbindung zur FIX-Anwendung auf dem Applikationsserver. Folgende Eingaben sind dabei unter FIX/Win möglich:

- *Drücken einer Taste* Die Umsetzung in ein FIX-Event erfolgt anhand einer Tastenbelegungstabelle, die vom Anwendungsentwickler konfigurierbar ist.
- *Anklicken von Elementen* FIX/Win sendet ein spezielles Event sowie die Mausposition von FIX-Objekten an FIX.
- *Anklicken eines Tasten-Labels* FIX-Anwendungen stellen in der Statuszeile die im aktuellen Kontext gültigen Tasten dar. FIX/Win erzeugt daraus spezielle Buttons, die Tasten-Labels genannt werden. Sie können angeklickt werden und senden das entsprechende FIX-Event zur Anwendung.
- *Anklicken eines Icons* FIX/Win stellt eine Box mit Icons zur Verfügung. Jedes der Icons ist mit einem oder mehreren FIX-Events belegt, die beim Anklicken zur FIX-Anwendung gesendet werden. Das Aussehen der Icons und die Eventbelegung kann vom Anwendungsentwickler unter FIX/Win eingestellt werden. Die Icons selbst können in einem nicht aktiven Zustand dargestellt werden; d.h. sie können in diesem Fall nicht angeklickt werden. Das Aktivieren und Deaktivieren von Icons übernimmt zum einen FIX selbst (soweit es Standardicons betrifft) und zum anderen die FIX-Anwendung.
- *Auswählen eines Menüpunktes* Die Anwendung kann ein Kontextmenü in FIX/Win öffnen. Bei der Auswahl eines Menüpunktes wird in FIX/Win eine vom Anwendungsentwickler definierte Aktion ausgeführt. In den meisten Fällen handelt es sich dabei um das Senden eines FIX-Events.

1. Wenn mit FIX für Windows gearbeitet wird, übernimmt diese Aufgabe der von NSI bereitgestellte Service `FXSRV`.

3.2 Elemente von FIX/Win

Ab der Version 3.0.0 arbeitet FIX/Win als MDI Anwendung. Damit ist es möglich, mehrere Verbindungen in der gleichen Programminstanz aufzubauen. Die Oberfläche von FIX/Win besitzt dazu die im Folgenden beschriebenen Elemente.

Hauptfenster

Dieses Fenster stellt den Rahmen der Anwendung FIX/Win dar. Es beinhaltet alle anderen Fenster. Das Menü von FIX/Win ist Bestandteil des Hauptfensters.

Client-Bereich

Der Client-Bereich ist der unmittelbar im Hauptfenster liegende Bereich. Er nimmt die anderen Fenster der MDI-Anwendung auf.

Container-Fenster

Ein Container-Fenster bildet den Rahmen für ein FIX/Win-Fenster. Der Client-Bereich kann mehrere Container-Fenster aufnehmen, die frei verschiebbar sind.

FIX/Win-Fenster

Das FIX/Win-Fenster bildet die Bedienoberfläche der FIX-Anwendung ab. Es wird in dem inneren Teil des Container-Fensters angeordnet. Der Begriff "Fenster" ist hier im Sinne der Programmierung von Windows zu verstehen, bei der ein Fenster ein rechteckiger Bereich ist, in dem Ausgaben vorgenommen werden. Ein Fenster muss dabei nicht unbedingt einen eigenen Rahmen besitzen und verschiebbar sein. Es kann Teil eines anderen Fensters sein.

Statuszeile

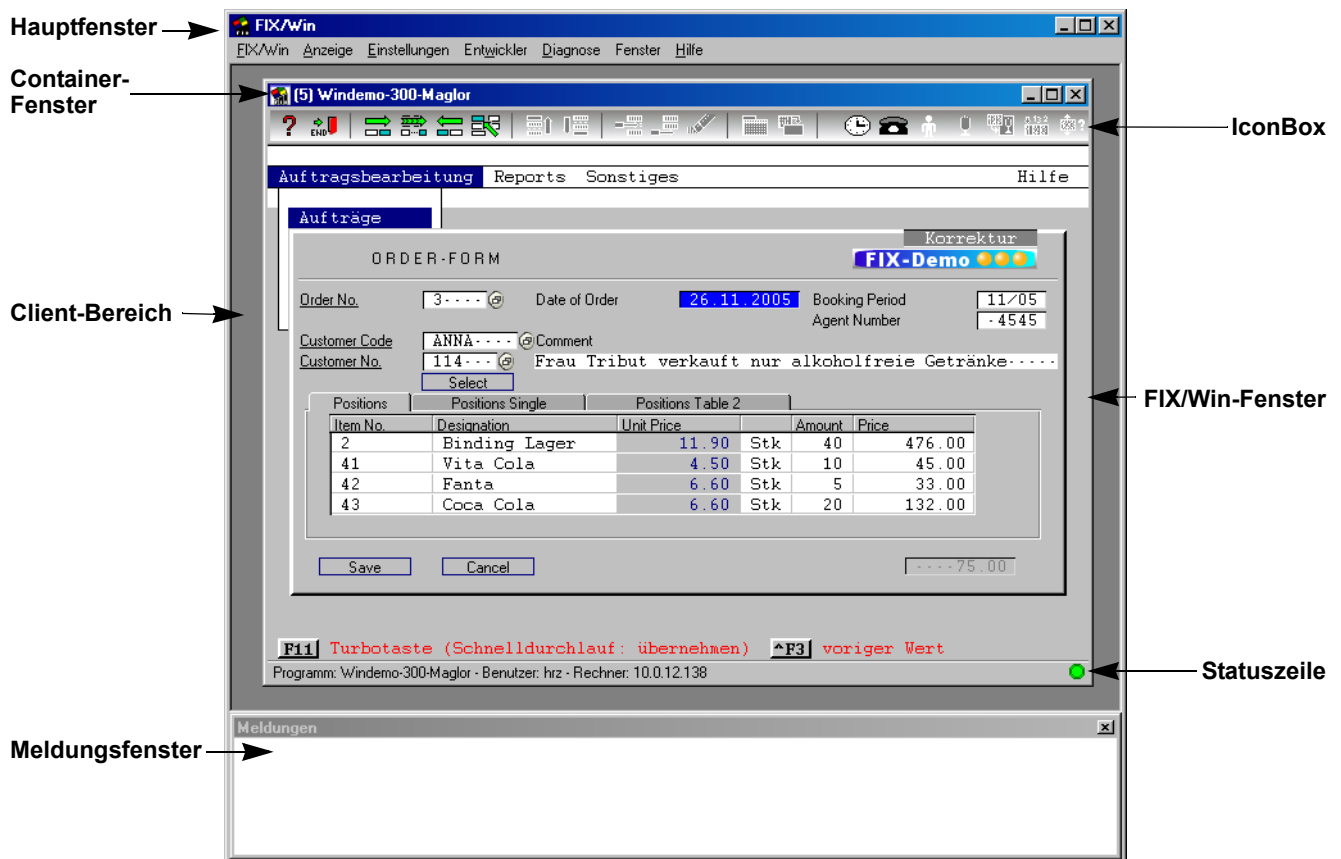
Die Statuszeile wird im unteren Teil eines Container-Fensters angeordnet. Sie zeigt Statusinformation zu der aktuellen Verbindung an.

IconBox

Die IconBox wird normalerweise im oberen Teil eines Container-Fensters angeordnet. Es besteht jedoch auch die Möglichkeit, sie im oberen Teil des Client-Bereichs zu platzieren.

Meldungsfenster

Im Meldungsfenster werden alle Meldungen von FIX/Win ausgegeben. Da eine Meldung einem der Container-Fenster zugeordnet werden muss, wird vor jeder Meldung der Titel des Fensters ausgegeben. Das Meldungsfenster wird normalerweise im unteren Teil des Client-Bereichs platziert. Durch das Verschieben des oberen Randes kann die Höhe des Meldungsfensters verändert werden. Als Option ist es möglich, das Meldungsfenster als eigenständiges Fenster außerhalb des Client-Bereichs zu platzieren



3.3 Versionsangabe von FIX/Win

Die Versionsangabe von FIX/Win ist in verschiedene Bereiche aufgeteilt, die bei bestimmten Änderungen angepasst werden:

- Die *Version* von FIX/Win: z.B. 3.0.0
- Das *Patchdate* von FIX/Win: z.B. 20061215
- Der *Featurelevel* von FIX/Win: z.B. 2

Version

Die Version von FIX/Win besteht aus einer dreistelligen Versionsnummer. Sie wird im Rahmen der Weiterentwicklung von FIX/Win hochgezählt. Je nach Änderungsumfang erfolgt die Änderung der Nummer an der ersten, der zweiten oder der dritten Stelle.

Patchdate

Das Patchdate enthält das Datum des Quellcodestandes, der für FIX/Win verwendet wurde, im Format YYYYMMDD. YYYY steht für das Jahr, MM für den Monat und DD für den Tag des Monats. Bei jeder Veränderung an FIX/Win die an Kunden ausgeliefert wird - sei es zur Fehlerbehebung oder zur Weiterentwicklung - wird das Patchdate auf das aktuelle Datum gesetzt. Aus dem Patchdate wird die Port-ID gebildet, indem der Wert um die Endung 10 ergänzt wird.

Featurelevel

Ab Version 2.9.5 Patchdate 20030929 wird von FIX und FIX/Win ein Featurelevel definiert. Das ist eine Nummer im Integerbereich, mit deren Hilfe ein FIX Programm erkennen kann, mit welcher FIX/Win Version es zusammenarbeitet

und mit der FIX/Win erkennen kann, mit welcher FIX Version es zusammenarbeitet. Dies erleichtert insbesondere Änderungen am Protokoll von kommenden Versionen. Dabei besitzen alle FIX und FIX/Win Versionen vor dem obigen Patchdate die Version 0. Hier ist also keine Differenzierung möglich. Der Stand vom 29.09.2003 besitzt die Version 1. Bei Änderungen am Protokoll, oder bei der Erweiterung um Features, die sowohl FIX als auch FIX/Win betreffen, wurde und wird der Featurelevel erhöht. Durch Auswertung dieser Version ist es FIX/Win und FIX möglich, die Features der jeweils anderen Seite zu nutzen oder deren Nutzung zu unterbinden, weil das Feature auf der Gegenseite nicht implementiert ist.

Mit der Version 3.0.0 wurde eine Änderung eingeführt, die es trotz der Verwendung des Featurelevels nicht möglich macht, dass eine ältere FIX Version mit FIX/Win zusammen arbeitet. FIX/Win 3.0.0 arbeitet deshalb nur mit FIX ab Version 3.0.0 zusammen und FIX 3.0.0 arbeitet nur mit FIX/Win ab Version 3.0.0 zusammen. Der Featurelevel wurde im Rahmen dieser Protokolländerungen (von 3) auf 4 erhöht.

Da mit der Version 3.1.0 die Unterstützung für Unicode eingeführt wurde, musste das Protokoll komplett überarbeitet werden. Aus diesem Grund kam es wieder zu einem Bruch in der Kompatibilität: FIX/Win 3.1.0 arbeitet nur mit FIX 3.1.0 (oder höher) zusammen und FIX 3.1.0 arbeitet nur mit FIX/Win 3.1.0 (oder höher) zusammen. Von FIX 3.1.0 werden zwei verschiedene Version ausgeliefert: FIX 3.1.0 Unicode und FIX 3.1.0. Von FIX/Win wird nur die Unicode-fähige Version ausgeliefert. Mit ihr können beide Versionen von FIX bedient werden. Im Rahmen dieser Umstellung wurde der Featurelevel von 4 auf 5 erhöht.

3.3.1 Ermitteln der Versionsangabe

Zur Ermittlung der Versionsangabe gibt es mehrere Möglichkeiten.

Lesen der Headerdatei

Die Headerdatei

```
include/VersionInfo.h
```

enthält die Versionsangabe, die beim Kompilieren der FIX/Win Version verwendet wurden. Die Datei kann in einem eigenen Framework zur Auswertung der Version verwendet werden. Folgende Makros werden definiert:

Tabelle 1: Versionsangaben in VersionInfo.h

Makro	Beschreibung
MAJORVERSION	Erste Stelle der Version
MINORVERSION	Zweite Stelle der Version
SUBVERSION	Dritte Stelle der Version
PATCHDATE	Patchdate
PRODUCTVER	Version als Zahl
STRPRODUCTVER	Version als Zeichenkette
FIXWIN_FEATURE_VERSION	Featurelevel

Menüpunkt "Hilfe/Info"

Dieser Menüpunkt öffnet eine Dialogbox mit der Versionsangabe. Hieraus können Version und Patchdate der einzelnen Komponenten entnommen werden.

Anzeige der Dateieigenschaften

In den Dateieigenschaften (ALT-Return im Windows-Explorer) im Reiter "Version" kann die Versionsinformation "Produktinformation" ausgelesen werden. Sie hat folgendes Format

```
<Version erste Stelle>,<Version zweite Stelle>, <Version dritte Stelle>,<Patchdate>
```

Die Versionsangabe ist in allen binären Dateien (Unterverzeichnis bin, alle .dll und .exe Dateien) vorhanden.

3.4 Anpassen und Erweitern von FIX/Win

Das Konzept von FIX/Win wurde so entworfen, dass Anpassungen und Erweiterungen durch eigene Programmierung möglich sind. FIX/Win bietet dazu drei verschiedene Möglichkeiten an, die miteinander kombiniert werden können:

- Die Konfiguration von FIX/Win kann angepasst werden.
- FIX/Win kann über Funktionen der Benutzerbibliothek erweitert werden.
- Der Kern von FIX/Win kann in ein eigenes Framework eingebunden werden.

Anpassen der Konfiguration

Die Einstellungen von FIX/Win werden in Konfigurationsdateien gehalten. Diese Dateien befinden sich unterhalb von zwei verschiedenen Verzeichnissen:

- dem *Arbeitsverzeichnis* von FIX/Win
- einem der Anwendung zugeordneten *Gruppenverzeichnis*

Das Arbeitsverzeichnis ist das Verzeichnis, in dem FIX/Win installiert wurde. Es enthält alle Dateien der Standardkonfiguration. Alle vom Anwendungsentwickler zusätzlich definierten Konfigurationen werden im Gruppenverzeichnis abgelegt. FIX/Win sucht seine Konfigurationsdateien beim Start einer Anwendung immer erst im Gruppenverzeichnis. Nur wenn die entsprechenden Konfigurationsdateien dort nicht zu finden sind, wird die Standardkonfiguration aus dem Arbeitsverzeichnis von FIX/Win verwendet. Durch dieses Konzept ist es möglich, zum einen FIX-Anwendungen mit der Standardkonfiguration zu starten, ohne zusätzliche neue Konfigurationen vornehmen zu müssen, und zum anderen FIX/Win für bestimmte Anwendungen den eigenen Bedürfnissen anzupassen, ohne die Standardkonfiguration zu verändern. Ein Gruppenverzeichnis kann von mehreren FIX-Anwendungen genutzt werden, so dass die Benutzer verschiedenster Programme mit den selben Einstellungen arbeiten. Eine Gruppe von FIX-Entwicklern kann sich also mit einem Gruppenverzeichnis eine eigene Umgebung für alle ihre Programme unter FIX/Win definieren.

Mit Hilfe dieser Anpassungen sind folgende Dinge möglich:

- Die Tasten können mit anderen FIX-Events belegt werden.
- Die Schriftart kann durch eine andere ersetzt werden.
- Die Farben für die verschiedenen FIX-Elemente können definiert werden.
- Zur Darstellung von Semigrafikzeichen können andere Bitmaps verwendet werden.
- Die Menge der Icons und die zu einem Zeitpunkt aktiven Icons können definiert werden.
- Die Bitmaps, die für die Icons verwendet werden, können definiert werden.

Die Tastenbelegung, die Schriftart, die Farben und die Bitmaps zur Darstellung der Semigrafikzeichen werden im Text dieses Handbuchs auch als Look&Feel bezeichnet. Eine genaue Beschreibung ist im Kapitel [Anpassung von FIX/Win auf Seite 41](#) zu finden.

Erweiterung über eine Benutzerbibliothek

FIX/Win bietet die Möglichkeit, beim Start den Pfad einer dynamischen Bibliothek (DLL) anzugeben. In dieser Bibliothek können Funktionen hinterlegt werden, die zu bestimmten Zeitpunkten von FIX/Win aufgerufen werden. Damit sind folgende Dinge möglich:

- Aufrufen einer beliebigen Windows-Funktion von der FIX-Anwendung aus.
- Bearbeiten von Daten, die von der FIX-Anwendung gesendet werden.
- Eigene Zeichenfunktionen für bestimmte Elemente des FIX-Bildschirms.

Eine genaue Beschreibung ist im Kapitel [Erstellen einer Benutzerbibliothek auf Seite 89](#) zu finden.

Erstellen eines eigenen Frameworks

Mit der Version 3.0.0 wurde FIX/Win auf C++ umgestellt und in zwei Teile aufgespalten:

- eine dynamische Bibliothek mit den Kernfunktionen von FIX/Win.

- ein Framework, das auf Basis dieser Bibliothek eine Anwendung bereitstellt.

Dadurch ist es möglich, die Bibliothek in ein eigenes Framework einzubetten und so FIX/Win in beliebigen Windows-Anwendungen zu nutzen. Die Bibliothek mit den Kernfunktionen wird jedoch erst mit einer neuen Version von FIX/Win freigegeben und dokumentiert, so dass diese Möglichkeit jetzt noch nicht zur Verfügung steht.

Die Datei `fixwin.exe`, die die alte FIX/Win Anwendung ersetzt und auf der Bibliothek mit den Kernfunktionen basiert wird im folgenden Text als Standard-Framework bezeichnet.

Da durch ein eigenes Framework viele Dinge an FIX/Win anders gestaltet werden können, beziehen sich die in dieser Dokumentation beschriebenen Funktions- und Verhaltensweisen nur auf das Standard-Framework. So ist beispielsweise der Ersteller eines Frameworks keinesfalls daran gebunden, die hier dokumentierten Kommandozeilenparameter zu verwenden. Das bedeutet, dass mit der Verwendung eines eigenen Frameworks Abweichungen zu der Dokumentation entstehen können.

4 Installation

Zur Installation von FIX/Win ist die Datei `FIXWin.exe` auszuführen. Sie entpackt und startet die Installation. Die Installation besteht aus mehreren Dialogen und ist im Wesentlichen selbsterklärend. Die folgenden Hinweise sind jedoch beim Installieren von FIX/Win hilfreich:

Allgemeine Hinweise

Vor dem Installieren sind laufende FIX/Win-Anwendungen zu beenden. Wenn FIX/Win bereits auf dem Rechner installiert wurde, dann ist diese Installation zu entfernen oder für die neue Installation ist ein anderes Verzeichnis zu wählen. Bei der Installation werden die Installationsinformationen einer bestehenden Version überschrieben. Die alte Version kann dann nicht mehr über den üblichen Mechanismus (über Systemsteuerung/Software) entfernt werden. Da FIX/Win abgesehen von den Installationsinformationen keine weiteren Informationen in der Registrierungsdatenbank von Windows ablegt, genügt es zum Entfernen einer solchen Version das alte Verzeichnis mit dem kompletten Inhalt zu löschen.

Abgesehen von diesen Einschränkungen ist es möglich, zwei oder mehrere verschiedene Versionen von FIX/Win auf einem Rechner zu installieren und zu starten.

Angabe von Benutzerinformationen

Neben dem Namen und dem Firmennamen ist in diesen Feldern die 9-stellige Seriennummer einzutragen, die auch zur Installation von FIX verwendet wird.

Angabe des Zielpfades

In diesem Feld ist ein Verzeichnis einzugeben, in das die Dateien von FIX/Win installiert werden sollen. Das Verzeichnis wird vom Installationsprogramm angelegt.

Angabe der Komponenten

Die Dateien von FIX/Win sind in mehrere Komponenten unterteilt:

- **Runtime:** Alle Dateien die zum Start von FIX/Win notwendig sind. Diese Komponente muss immer installiert werden.
- **Development:** Alle Dateien, die zum Entwickeln mit FIX/Win notwendig sind. Diese Komponente muss nur installiert werden, wenn Anpassungen an FIX/Win vorgenommen werden sollen.
- **Beispielgruppe nsigrp:** Diese Dateien werden zum Start der Beispielanwendung `windemo` benötigt. Diese Komponente muss dann installiert werden, wenn die Anwendung `windemo` gestartet werden soll.
- **Beispiel Bibliothek:** Alle Quellcodes zur Erstellung der Benutzerbibliothek für die Beispielanwendung `windemo`. Diese Komponente muss dann installiert werden, wenn die Anwendung `windemo` angepasst werden soll oder wenn ein Beispiel als Vorlage für eine Benutzerbibliothek benötigt wird.

C-Laufzeitbibliotheken

FIX/Win wurde mit Microsoft Visual Studio 2005 entwickelt. Deshalb sind zum Start die entsprechenden C-Laufzeitbibliotheken notwendig. Auf vielen Rechnern sind diese bereits installiert. Trifft dies nicht zu, dann sind die Dateien aus dem Unterverzeichnis `cruntime` in das Unterverzeichnis `bin` oder in das Systemverzeichnis (in der Regel `C:\windows\system32`) zu kopieren.

5 Verzeichnisaufbau und Komponenten von FIX/Win

Das Arbeitsverzeichnis von FIX/Win enthält nach der Installation folgende Verzeichnisse:

bin

Dieses Verzeichnis enthält alle ausführbaren Dateien (DLLs zählen zu den ausführbaren Dateien) von FIX/Win:

- `Fixwin.exe`: Das Standard-Framework.
- `FWSimple.exe`: Ein Beispiel-Framework.
- `FixwinCore.dll`: Die Kern-Klassen von FIX/Win.
- `ResFixwin.dll`: Ressourcen zu dem Standard-Framework.
- `ResFixwinCore.dll`: Ressourcen zu den Kern-Klassen.
- `bmpack.exe`: Ein Programm zum Packen von Bitmaps.
- `crypt.exe`: Ein Programm zum Verschlüsseln von Passwörtern
- `edprg.exe`: Ein Programm zum Bearbeiten der Programmtabelle

config

Das Verzeichnis enthält alle Konfigurationsdateien von FIX/Win. Diese Dateien können durch die Dateien des Gruppenverzeichnisses überlagert werden. Die einzelnen Dateien sind in dem Abschnitt [Aufbau eines Gruppenverzeichnisses auf Seite 41](#) beschrieben.

LookAndFeel

Jedes Unterverzeichnis dieses Verzeichnisses enthält alle Dateien, die ein Look&Feel beschreiben. Dazu zählen im Wesentlichen die Bitmaps für Semigrafikzeichen, Rahmen und Tasten, die Farbeinstellungen, Fonts und Einstellungen der IconBox. Die Dateien können von Dateien des Gruppenverzeichnisses überlagert werden. Die einzelnen Dateien sind in dem Abschnitt [Aufbau eines Gruppenverzeichnisses auf Seite 41](#) beschrieben.

LookAndFeel-src

Die gepackten Bitmapdateien werden aus einzelnen Bitmaps zusammengesetzt. Diese Bitmaps werden im Verzeichnis `LookAndFeel-src` abgelegt. Die einzelnen Dateien sind in dem Abschnitt [Aufbau eines Gruppenverzeichnisses auf Seite 41](#) beschrieben.

bpckcfg

Dieses Verzeichnis enthält die Konfigurationsdateien des Programms `bmpack`. Sie beschreiben die Bitmap-Sätze, die gepackt werden sollen.

nsigrp

`nsigrp` ist ein Beispiel für ein Gruppenverzeichnis. Es wird von der FIX-Beispielanwendung `windemo` verwendet.

include

Dieses Verzeichnis enthält die Header-Dateien, die zur Entwicklung eines eigenen FIX/Win-Frameworks oder zur Erstellung einer eigenen Benutzer-DLL notwendig sind:

- `FixwinCore.h`: Enthält alle Klassen- und Funktionsdefinitionen des FIX/Win Kerns.
- `fwown.h`: Enthält alle Definitionen, die für die Erstellung einer eigenen Benutzerbibliothek notwendig sind und den Kern von FIX/Win betreffen.
- `fwownStd.h`: Enthält alle Definitionen, die für die Erstellung einer eigenen Benutzerbibliothek notwendig sind und das Standard-Framework von FIX/Win betreffen.
- `VersionInfo.h`: Enthält Definitionen, die die Version von FIX/Win betreffen.

lib

Das Verzeichnis `lib` enthält die Bibliothek `FixwinCore.lib`. Diese Bibliothek muss zusammen mit einem eigenen Framework gebunden werden, damit die Klassen und Funktionen des FIX/Win Kerns benutzt werden können.

src/ResFixwin

In diesem Verzeichnis liegt ein Visual Studio Projekt zur Erstellung der Datei `ResFixwin.dll`. In dieser Datei sind alle Windows-Ressourcen enthalten, die vom Standard-Framework benötigt werden. Zur Anpassung an eine andere Sprache oder an andere Gegebenheiten können die Dateien dieses Verzeichnisses verändert und neu kompiliert werden.

src/ResFixwinCore

Dieses Verzeichnis enthält ein Visual Studio Projekt zur Erstellung der Datei `ResFixwinCore.dll`. In dieser Datei sind alle Windows-Ressourcen enthalten, die von den Kernklassen benötigt werden. Zur Anpassung an eine andere Sprache oder an andere Gegebenheiten können die Dateien dieses Verzeichnisses verändert und neu kompiliert werden.

src/FWSimple

Die Dateien dieses Verzeichnisses bilden ein einfaches Framework, das den FIX/Win Kern verwendet. Der Quellcode kann als Beispiel oder als Basis für eigene Entwicklungen verwendet werden. Obwohl das Interface der Kernklassen in der dieser Version noch nicht freigegeben und dokumentiert wird, wird das Beispiel schon mit ausgeliefert. Damit soll ein Vorgeschmack auf das Entwickeln von Anwendungen mit den FIX/Win Kernklassen gegeben werden.

6 Grundlegende Konfiguration

6.1 Kommandozeilenparameter

Bei der Installation wird für das Standard-Framework ein Punkt im Startmenü angelegt. Dabei wird als Befehlszeile `fixwin.exe` und als Arbeitsverzeichnis das Verzeichnis angegeben, in dem FIX/Win installiert wurde. Dieses Verzeichnis ist für FIX/Win die Grundlage beim Auffinden von Konfigurationsdateien.

Innerhalb der Befehlszeile können die im folgenden beschriebenen Kommandozeilenparameter angegeben werden.

/ptab <Path>

Wenn dieser Parameter angegeben wird, dann wird die Programmtabelle von FIX/Win aus der Datei <Path> gelesen. Fehler der Parameter, dann wird die Datei `program.fix` aus dem Arbeitsverzeichnis verwendet.

/pgm <Programm>

Dieser Parameter definiert den Wert <Programm> als Name des Programm aus der Programmtabelle zu dem eine Verbindung aufgebaut werden soll. Wenn der Schalter nicht angegeben wird, dann muss der Name aus dem Dialog beim Verbindungsaufbau ausgewählt werden.

/user <User>

Der Wert <User> wird beim Verbindungsaufbau für den Namen des Benutzers verwendet. Wenn der Kommandozeilenparameter nicht angegeben wird, dann wird der Inhalt der Umgebungsvariablen `USER` oder `LOGNAME` verwendet.

/pass <Password>

Dieser Parameter definiert <Password> als das Passwort, das beim Verbindungsaufbau verwendet wird. Wenn der Parameter nicht angegeben wird, dann muss das Passwort beim Dialog zum Verbindungsaufbau eingegeben werden.

/cryptpass

Die Angabe dieses Parameters bestimmt, dass das Passwort verschlüsselt angegeben wurde oder beim Dialog zum Verbindungsaufbau verschlüsselt eingegeben werden muss. Zum Verschlüsseln eines Passworts kann das Programm `crypt.exe` verwendet werden.

/xtracmd <args>

Die Angabe dieses Parameters bewirkt, dass die Zeichenkette <args> als Parameter an das Startscript übergeben wird. Der in <args> enthaltene Parameter darf die Zeichen `\` oder `/` nicht enthalten. Müssen derartige Zeichen verwendet werden, dann ist eine Ersatzdarstellung zu verwenden. Eine Interpretation des Parameters und eine Auflösung der Ersatzdarstellung bleibt dem Startscript oder dem von ihm aufgerufenen FIX-Programm überlassen. Bei der Verwendung von Leerzeichen ist <args> in doppelte Anführungszeichen einzufassen.

/nostdoutput

Die Ausgabe auf dem Standardkanal (`stdout`) wird vor dem Start eines FIX-Programmes im Meldungsfenster angezeigt. So sind dort z.B. alle Meldungen des Startscripts zu sehen. Durch die Angabe dieses Schalters läßt sich die Ausgabe unterdrücken.

/auto

Die Angabe dieses Schalters bewirkt, dass unmittelbar nach dem Start von FIX/Win ein Verbindungsaufbau gestartet wird. Wenn nicht alle notwendigen Parameter (*/pgm /user /pass*) angegeben wurden, dann werden die restlichen Parameter in dem Dialog zum Verbindungsaufbau abgefragt. Schlägt der Verbindungsaufbau fehl, dann wird ebenfalls der Dialog zum Verbindungsaufbau gestartet und der Benutzer kann die Werte verändern oder FIX/Win abbrechen.

Darüber hinaus hat die Angabe des Parameters */auto* folgende Auswirkungen, wenn FIX/Win zusätzlich mit dem Parameter */single* gestartet wurde:

- Das Hauptfenster wird erst nach dem Verbindungsaufbau eingeblendet.
- FIX/Win wird nach dem Verbindungsabbau automatisch beendet.

/logmsg <Logdatei>

Dieser Parameter definiert die Datei *<Logdatei>* als Logdatei. Alle Ausgaben, die in das Meldungsfenster geschrieben werden, werden auch in diese Datei geschrieben.

/msgwinext

Die Angabe dieses Parameters bewirkt, dass das Meldungsfenster zu Beginn als externes Fenster dargestellt wird.

/hide

Die Angabe dieses Parameters bewirkt, dass das FIX/Win Fenster nicht angezeigt wird.

/dev

Bei der Angabe dieses Parameters wird das Menü "Entwickler" aktiviert. Eine Beschreibung dieses Menüs ist in [Bedienung durch das Entwicklermenü auf Seite 75](#) zu finden.

/diag <n>

Bei der Angabe dieses Parameters wird das Menü "Diagnose" aktiviert. Damit ist es möglich, bestimmte Diagnosen zur Fehlersuche in FIX-Anwendungen und in FIX/Win zu erstellen. Eine genaue Beschreibung ist in [Möglichkeiten des Diagnosemenüs auf Seite 77](#) zu finden.

/protoprof

Dieser Parameter bewirkt die Erzeugung einer Logdatei zur Analyse des Protokolls zwischen FIX und FIX/Win. Eine Beschreibung dieses Features ist in dem Abschnitt [Erstellen von Protokoll-Profilen auf Seite 108](#) zu finden.

/frown <Path>

Durch die Angabe dieses Parameters lädt FIX/Win die Benutzerbibliothek *<Path>*. Die Erstellung einer Benutzerbibliothek wird in dem Kapitel [Erstellen einer Benutzerbibliothek auf Seite 89](#) beschrieben.

/resfw <Path>

/resfwc <Path>

Mit Hilfe dieser Parameter kann die Verwendung von bestimmten Bibliotheken mit Ressourcen erreicht werden. Der Parameter */resfw* definiert die Datei *<Path>* als Bibliothek mit Ressourcen für das Standard-Framework. Wird der

Parameter nicht angegeben, dann wird die Datei `ResFixwin.dll` verwendet. Der Parameter `/resfwc` definiert die Datei `<Path>` als Bibliothek mit Ressourcen für die Kern-Klassen. Wird der Parameter nicht angegeben, dann wird die Datei `ResFixwinCore.dll` verwendet.

Der Quellcode der beiden Bibliotheken liegt in jeweils einem Unterverzeichnis von `src`. Er kann als Vorlage zur Erstellung von eigenen Bibliotheken genutzt werden.

```
/grpdir <Path>
/style <nr>
/size <nr>
/oneiconbox
```

Die Angabe des Parameters `/grpdir` bewirkt, dass alle Verbindungen der MDI Oberfläche das gleiche Gruppenverzeichnis `<Path>` verwenden. Das hat folgende Auswirkungen:

- Die Angabe des Gruppenverzeichnisses in der Programmtabelle wird ignoriert.
- Die Dateien eines Look&Feels werden nur einmal geladen werden, was Speicherplatz und Plattenzugriffe beim Start von mehreren FIX-Anwendungen spart.
- Bei der Umschaltung von Größe und Stil, wird die Größe und der Stil aller FIX/Win-Fenster geändert, die unter dieser Programminstanz von FIX/Win geöffnet wurden.
- Der in der Ressource-Datei zu einer Anwendung gespeicherte Stil und die Größe werden nach dem Verbindungsaufbau nicht gesetzt, weil das Setzen sich auch auf die anderen FIX-Anwendungen auswirkt. Statt dessen kann der anfänglich verwendete Bildstil und die Bildgröße durch die Angabe der Parameter `/style` und `/size` bestimmt werden.
- Die IconBox-Definitionsdatei muss die Iconlisten und Iconsets von allen Anwendungen definieren, die in dieser Programminstanz geöffnet werden.

Zusätzlich ist es bei der Verwendung von `/grpdir` möglich, den Parameter `/oneiconbox` anzugeben. In diesem Fall besitzt ein Container-Fenster keine eigene IconBox. Statt dessen wird eine IconBox in den Client-Bereich des Hauptfensters eingebunden. Die Umschaltung dieser IconBox erfolgt mit dem Wechsel auf ein anderes FIX/Container-Fenster.

```
/userdir <Path>
```

Dieser Parameter legt `<Path>` als Benutzerverzeichnis fest. Das Benutzerverzeichnis wird zum Auffinden von Ressourcen benutzt (siehe [Ressourcendateien von FIX/Win auf Seite 80](#)).

Darüber hinaus kann es in einer Benutzerbibliothek für anwendungseigene Zwecke verwendet werden (siehe [GetUserDir - Pfad des Benutzerverzeichnisses ermitteln auf Seite 101](#)).

```
/single
```

Wenn dieser Parameter angegeben wird, verhält sich FIX/Win wie eine SDI (Single Document Interface) Anwendung.

```
/usemdikeys
```

Wenn dieser Parameter angegeben wird, dann werden die Tasten zur Steuerung einer MDI Anwendung (CTRL-F4, CTRL-F6, ...) verwendet. Wird der Parameter nicht angegeben, dann wird beim Drücken dieser Tasten versucht, anhand der Tastaturliste ein Event zu ermitteln und an die FIX-Anwendung zu senden.

6.2 Globale Einstellungen von FIX/Win

In der Datei

```
config/fixwin.frc
```

können allgemeine Einstellungen zu FIX/Win vorgenommen werden. Diese Datei wird bereits beim Start von FIX/Win geladen. Damit sind die Einstellungen unabhängig von einer bestimmten Verbindung. Die Einstellungen sind im FIX/Win-Ressourceformat vorzunehmen (siehe [Allgemeines zu Ressourcen auf Seite 79](#)). Folgende Einstellungen können verwendet werden:

MsgwinFont

Name der Schriftart, die für das Meldungsfenster verwendet wird.

Der Default für diese Ressource hängt vom Windows-Gebietsschema ab. In einer japanischen Umgebung wird die Schriftart *MS PGothic* verwendet. In allen anderen Umgebungen wird die Schriftart *MS Sans Serif* verwendet.

MsgwinFontSize

Größe der Schrift, die für das Meldungsfenster verwendet wird.

Wird der Wert nicht angegeben, dann wird der Wert 13 verwendet.

MsgwinLines

Anzahl der dargestellten Zeilen im Meldungsfenster.

Die Anzahl der Zeilen kann zur Laufzeit durch das Ändern der Fenstergröße verändert werden. Diese Ressource definiert nur den Anfangswert. Wenn der Wert nicht angegeben wird, dann wird der Wert 8 verwendet.

StatuslineFont

Name der Schriftart, die für die Statuszeile verwendet wird.

Der Default für diese Ressource hängt vom Windows-Gebietsschema ab. In einer japanischen Umgebung wird die Schriftart *MS PGothic* verwendet. In allen anderen Umgebungen wird die Schriftart *MS Sans Serif* verwendet.

StatuslineFontSize

Größe der Schrift, die für die Statuszeile verwendet wird.

Wird der Wert nicht angegeben, dann wird der Wert 14 verwendet.

StatuslineHeight

Höhe der Statuszeile.

Bei einer Vergrößerung der Schrift kann es notwendig sein, den Standardwert von 18 auf einen größeren Wert zu setzen.

Weitere

Die folgenden Einstellungen können auch durch den gleichnamigen Kommandozeilenparameter bestimmt werden. Die Bedeutung ist bei dem jeweiligen Kommandozeilenparameter nachzulesen. Kommt es dadurch zu Konflikten, dann gilt die Angabe des Kommandozeilenparameters.

- logmsg
- single
- cryptpass
- oneiconbox

- msgwinext
- usemdikeys
- nostdoutput

Bis auf `logmsg` ist für alle Einstellungen der Wert `true` oder `false` anzugeben, wobei als Default der Wert `false` verwendet wird.

6.3 Einrichten von FIX-Anwendungen

Sämtliche FIX-Anwendungen, die aus FIX/Win gestartet werden sollen, müssen in die Programmtabelle eingetragen werden. Weiterhin muss auf dem Applikationsserver für jedes FIX-Programm ein Startskript erstellt werden.

6.3.1 Eintragen des Programmes

Hierzu ist die Programmtabelle zu erweitern. Sie wird entweder aus der Datei `program.fix` oder aus der Datei, die mit dem Kommandozeilenparameter `/ptab` übergeben wird, gelesen. In dieser Datei ist für jede FIX-Anwendung eine Zeile mit folgendem Aufbau einzutragen:

<Programm>, <Gruppe>, <Res-Datei>, <X>, <Y>, <Cmd>, <Hostname>

Dabei haben die einzelnen Komponenten folgende Bedeutung:

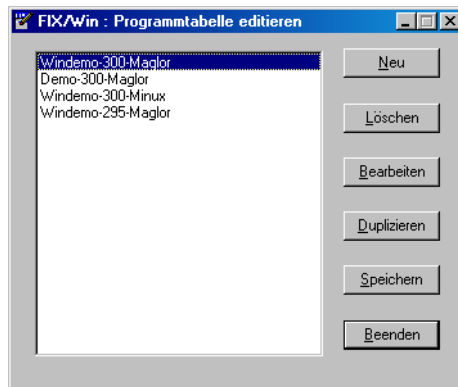
- <Programm> ist der Name, unter dem die FIX-Anwendung später in einer Auswahlbox angewählt werden kann oder welcher mit dem Schalter `/pgm` übergeben wird. Der Name muss in der Programmtabelle eindeutig sein.
- <Gruppe> ist der absolute Pfad des Gruppenverzeichnisses, in dem vom Standard abweichende Konfigurationsdateien abgelegt sind.
- <Res-Datei> bezeichnet eine Ressourcendatei, in der alle vom Benutzer einstellbaren Optionen (siehe [Menüpunkt "Übersicht" auf Seite 38](#)) einer Anwendung abgelegt werden.
- <X>, <Y> definiert die zur Verfügung stehende Größe des FIX/Win-Bildschirms in Zellen. Üblicherweise werden hier die Werte 80,25 eingetragen. Bei einer Erhöhung der Werte müssen die Masken der FIX-Anwendung verändert werden, so dass diese den größeren Platz auch nutzen.
- <Cmd> bezeichnet ein Startskript (siehe nächster Abschnitt) auf Seite der Anwendung, das eine Ablaufumgebung aufbaut und die Anwendung startet.
- <Hostname> ist der Hostname des Rechners, auf dem das Startskript liegt. Alternativ kann hier auch eine IP-Adresse angegeben werden. In beiden Fällen ist es möglich, den Port anzugeben, an den sich FIX/Win verbinden soll. Dazu ist die Nummer des Ports durch ":" getrennt hinter den Hostnamen oder die IP-Adresse zu schreiben. Fehlt die Angabe, dann wird der Port verwendet, der dem Service `exec` zugeordnet wurde (in der Regel 512).

Kommentare können - wie in allen anderen Konfigurationsdateien auch - mit einem `"#"` eingeleitet werden, wobei das Zeilenende das Ende des Kommentars bildet.

6.3.2 Bearbeiten der Programmtabelle mit "edprg"

Mit dem Programm `edprg` kann die Programmtabelle über einen interaktiven Dialog editiert werden. Standardmäßig wird die Programmtabelle `program.fix` aus dem Arbeitsverzeichnis von FIX/Win `edprg` als Parameter mitgegeben. Durch Verändern der Eigenschaften des Icons im Programm-Manager (siehe Windows-Handbuch) kann hier auch eine eigene Datei als Programmtabelle angegeben werden. Dabei ist zu beachten, dass diese Tabelle FIX/Win durch den `/ptab`-Schalter mitgegeben werden muss.

Nach dem Start von `edprg` wird eine Dialogbox eingeblendet, in der alle Programme der Programmtabelle aufgelistet sind. Ein Element der Liste, der *aktuelle Eintrag*, ist immer ausgewählt. Neben dieser Liste befindet sich eine Reihe von Buttons, mit denen die Liste verändert werden kann.



Folgende Aktionen können durchgeführt werden:

Neue Programme in die Programmtabelle aufnehmen

Hierzu ist "Neu" anzuklicken. Es wird eine weitere Dialogbox mit den Buttons "OK" und "Abbruch" eingeblendet, in der alle Parameter des Programms einzutragen sind. Einige Werte sind mit Standardvorgaben belegt. Diese können übernommen werden, wenn mit den Grundeinstellungen von FIX/Win gearbeitet werden soll, oder mit eigenen Angaben überschrieben werden. Der neu angelegte Eintrag wird hinter dem aktuellen Eintrag eingefügt. Durch Anklicken von "OK" wird der Eintrag neu angelegt. Durch Anklicken von "Abbruch" kann der Vorgang der Neuanlage abgebrochen werden. Man sollte darauf achten, dass der Programmname in der Programmtabelle eindeutig ist.

Programme aus der Programmtabelle löschen

Durch Anklicken von "Löschen" wird der aktuelle Eintrag aus der Liste gelöscht.

Programme der Programmtabelle bearbeiten

Hierzu ist "Bearbeiten" anzuklicken. Die Werte des aktuellen Eintrags werden in eine weitere Dialogbox mit den Buttons "OK" und "Abbruch" zur Bearbeitung geladen. Die Übernahme geänderter Werte erfolgt durch Anklicken von "OK" in der Dialogbox, der Abbruch der Bearbeitung durch Anklicken von "Abbruch".

Programme duplizieren

Dies kann durch Anklicken von "Duplizieren" erreicht werden. Die Werte des aktuellen Eintrags werden kopiert und hinter ihm eingefügt. Damit der Programmname eindeutig ist, wird eine Tilde ("~") angehängt, wenn es die Programmnamenlänge zulässt. Gelingt das Anhängen der Tilde nicht, wird der neue Eintrag abgelehnt.

Geänderte Programmtabelle speichern

Durch Anklicken von "Speichern" wird die Programmtabelle in der entsprechenden Datei abgelegt.

Nachdem der Editiervorgang beendet ist, kann das Programm `edprog` durch Anklicken von "Ok" beendet werden. Sollten Änderungen an der Programmtabelle vorgenommen worden sein, die nicht gespeichert wurden, macht das Programm darauf aufmerksam, und die betreffenden Änderungen können gespeichert oder verworfen werden.

6.3.3 Erstellen eines Startskripts

Dieses Skript dient zum Start der FIX-Anwendung und ist auf dem Applikationsserver anzulegen. Für das Startskript kann in der Programmtabelle entweder ein absoluter Pfad oder ein Pfad relativ zum HOME-Verzeichnis eines Benut-

zers eingetragen werden. Im zweiten Fall kann das Startskript nur gefunden und ausgeführt werden, wenn man beim Verbindungsaufbau den Namen dieses Benutzers verwendet.

Das Skript sollte folgende Arbeitsschritte vornehmen:

- Eine Ablaufumgebung für die Anwendung aufbauen.
- Die Anwendung selbst starten. Hierfür muss der Schalter `-service` angegeben werden, um der Anwendung mitzuteilen, dass sie von einem Frontend bedient wird.

Hier ein Beispielskript, das als Grundlage zur Erstellung eigener Skripts verwendet werden kann:

```

:
# Beispiel-Skript, das die Demo-Anwendung unter FIX/Win startet

# Umgebungsvariablen FXDIR und FXDIRSYS setzen
FXDIR=/usr/fix/fix
FXDIRSYS=/usr/fix/sys
export FXDIR FXDIRSYS
# Warnungen von etc/stdprofile aufgrund fehlender Bildschirmbeschreibung unterdrücken
FXTERMPATH=""
export FXTERMPATH

# zum Verzeichnis der Anwendung wechseln
cd $FXDIR/demo
# .profile ausführen und so Umgebung der Anwendung aufbauen
. .profile
# Anwendung starten
exec $FXHOMESYS/bin/demo -service

```

FXTERMPATH wird auf einen Leerstring gesetzt, um Warnmeldungen beim Einschleusen von `$FXDIR/etc/std-profile` bezüglich der Terminaleinstellung zu verhindern. Danach wird in das Verzeichnis der entsprechenden FIX-Anwendung gewechselt und dort `.profile` ausgeführt, wodurch die Umgebung für die Anwendung aufgebaut wird. Zum Schluß wird die Anwendung mit dem Schalter `-service` gestartet, der dem FIX-Programm anzeigt, dass es von einem Frontend bedient wird. Näheres hierzu ist im FIX-Handbuch zu finden.

Hierzu ist noch anzumerken, dass das Startskript sowie alle von der FIX-Anwendung direkt oder indirekt aufgerufenen Programme keine interaktiven Abfragen enthalten dürfen. FIX/Win ist speziell auf FIX-Anwendungen zugeschnitten und kann keine Eingabe an andere Programme weiterleiten. Wohl aber ist es diesen Programmen gestattet, Ausgaben vorzunehmen, die auf FIX/Win-Seite in eine Datei geschrieben werden (siehe hierzu [Ausgabe von Listen auf Seite 39](#)).

Verbindet sich FIX/Win an ein FIX-Programm für Windows, dann ist als Startskript eine Stapeldatei (*.cmd) zu verwenden, die die gleichen Aufgaben erfüllt

Ausgabe von Startmeldungen

Um bereits vor dem Start des FIX-Programms den Benutzer über gewisse Dinge zu informieren oder zur Ausgabe von Fehlern, welche bereits im Startscript auftreten, kann folgendes Feature verwendet werden.

Erfolgt in einem Startscript eine Textausgabe auf `stderr`, die mit dem Text

```
MsgBox:
```

beginnt, dann wird der Rest des Textes bis zum ersten Zeilenumbruch von *FIX/Win* in einer Windows-Messagebox ausgegeben, die mit dem OK-Button bestätigt werden muss. Die gesamte Länge der Ausgabezeile darf dabei jedoch 160 Zeichen nicht überschreiten.

Beispiel

```
echo "MsgBox: Diese Anwendung kann zur Zeit nicht gestartet werden" >&2
exit
```

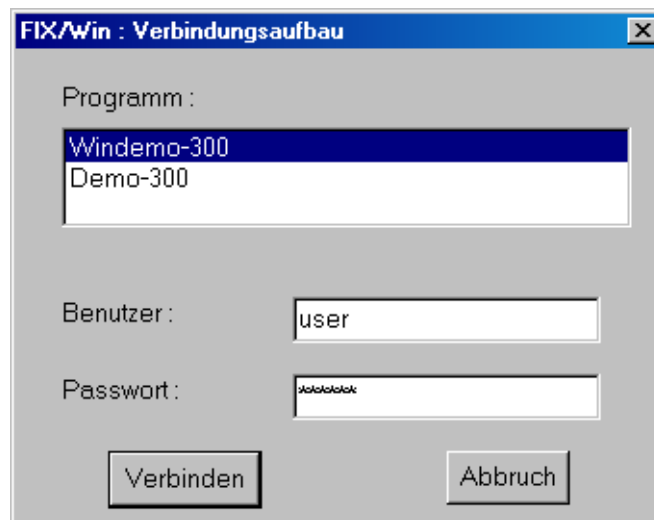
Dieses Startscript gibt eine Meldung in einer Messagebox aus und beendet sich. Es kann das Originalstartscript ersetzen, während z.B. eine neue Version der Anwendung eingespielt wird.

2 Arbeiten mit FIX/Win

Nachdem FIX/Win installiert wurde und die Anwendungen eingerichtet wurden, kann FIX/Win durch Anklicken im Startmenü gestartet werden.

Nach dem Start kann mit dem Menüpunkt `FIX/Win/Verbindungsaufbau` die Dialogbox zum Verbindungsaufbau gestartet werden. Dort kann die gewünschte FIX-Anwendung aus einer Liste ausgewählt werden.

Abb. 1 Dialog zum Verbindungsaufbau



In den beiden unteren Feldern werden Benutzer und Passwort zur Anmeldung auf dem Applikationsserver eingegeben. Der Benutzername wird aus den Umgebungsvariablen `USER` oder `LOGNAME` vorbesetzt. Aus Sicherheitsgründen wird das Passwort nicht im Klartext angezeigt, sondern jedes Zeichen wird durch einen "*" angezeigt. Die Verbindung mit dem Hostrechner wird aufgebaut, wenn "Verbinden" angeklickt wird. Durch Klicken auf "Abbrechen" wird der Dialog ohne einen Verbindungsaufbau beendet.

1 Bedienung mit der Tastatur

Die FIX-Anwendung kann unter FIX/Win mit der Tastatur und mit der Maus bedient werden.

Dabei wird standardmäßig die in Tabelle 2 dargestellte Belegung der Tasten verwendet. Diese Belegung kann jedoch durch Anlegen einer eigenen Tastenbelegung geändert werden¹. Es wird von einer deutschen PC-Tastatur ausgegangen. Bei den Tasten, die Beschriftungen haben, ist diese in “” angegeben.

Tabelle 2: Standardtastenbelegung

PC-Taste	Normal	Shift	Strg	Shift+Strg
“Ende”	EN	-	-	-
“Pos1”	kh	-	-	-
“Einfg”	CI	-	-	-
“Entf”	CD	-	-	-
Pfeil links	kl	PL	-	-
Pfeil rechts	kr	PR	-	-
Pfeil hoch	ku	-	-	-
Pfeil runter	kd	-	-	-
Bild hoch	PU	-	-	-
Bild runter	PD	-	-	-
Backspace	DL	-	-	-
Return	RT	-	-	-
Tab	TB	BT	-	-
“F1”	HP	f1	-	h1
“F2”	f2	BT	WI	h2
“F3”	f3	g3	LI	h3
“F4”	f4	g4	CD	h4
“F5”	f5	g5	WD	h5
“F6”	f6	g6	LD	h6
“F7”	f7	g7	CE	h7
“F8”	f8	g8	ST	h8
“F9”	f9	g9	-	h9
“F10”	f0	g0	PT	h0
“F11”	g1	-	-	-
“F12”	MD	g2	-	-
rechte Maustaste	f9	RT	kh	EN
linke Maustaste	-	RT	-	-
mittlere Maustaste	EN	-	-	-

1. siehe [Abschnitt 4 auf Seite 53](#)

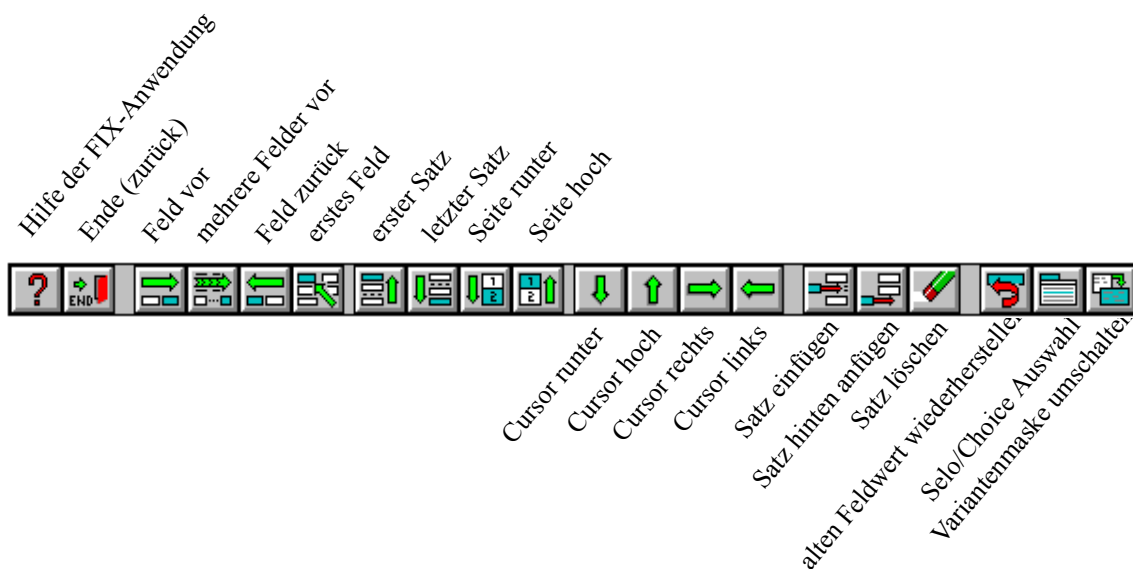
2 Arbeiten mit der Maus

2.1 Die IconBox

Im Gegensatz zum Terminalbetrieb kann unter FIX/Win auch die Maus zur Bedienung eines FIX-Programms eingesetzt werden. Zum einen sind Menüpunkte, Selo- oder Choice-Datensätze, Felder¹ und Tasten-Labels mit der linken Maustaste anklickbar. Zum anderen kann die rechte und die mittlere Maustaste² mit einem FIX-Event belegt (z.B. *EN*) sein. Bei einem Doppelklick mit der Maus werden zwei Tasten zur FIX-Anwendung gesendet: beim ersten Klick die normal belegte Taste und beim zweiten die Taste mit der Shift-Kombination. Wird zusätzlich zu einem Doppelklick noch die Strg-Taste gedrückt, dann wird als erstes die Taste in der Spalte Strg gesendet, gefolgt von der Taste in der Spalte Shift+Strg. So sendet beispielsweise ein Doppelklick mit der rechten Maustaste *f9*, gefolgt von *RT*, eine in FIX-Anwendungen häufig benötigte Kombination.

Desweiteren kann mit der Maus ein Icon der IconBox angeklickt werden. Auch hierdurch wird ein Event zur FIX-Anwendung gesendet. Da der Inhalt der IconBox ebenfalls, wie die Tastenbelegung, vom Anwendungsentwickler frei definierbar ist, soll an dieser Stelle eine Erläuterung der Standard-IconBox genügen. Sie bezieht sich hauptsächlich auf die Bearbeitung von Masken. Ihr Aufbau ist folgender Grafik zu entnehmen:

Abb. 2 FIX/Win Standard-IconBox



2.2 Mauszeiger unter FIX/Win

FIX/Win symbolisiert seinen Zustand durch den Mauszeiger. Folgende Symbole sind dabei möglich:



FIX/Win ist im *Bereitschafts-Modus*. Es kann mit der Maus geklickt werden und Zeichen können über die Tastatur eingegeben werden.



FIX/Win befindet sich im *Warte-Modus*. Es sollten keine Eingaben getätigt werden, da FIX/Win mit anderen Dingen (Laden der Konfiguration, Bildaufbau, usw.) beschäftigt ist.

1. Ab FIX Version 2.9.3 sind Felder anklickbar, wenn die Anwendung es vorsieht.

2. Bei Systemen mit nur zwei Maustasten kann die mittlere Maustaste simuliert werden, indem die rechte und dann die linke Maustaste gedrückt wird, ohne vorher die rechte loszulassen.



FIX/Win befindet sich im *Busy*-Modus. Im Gegensatz zum *Warte*-Modus können Tastatureingaben getätigt werden, da FIX/Win selbst nicht beschäftigt ist, sondern auf die FIX-Anwendung wartet. Mauseingaben werden jedoch nicht angenommen, da der Bildschirm erst aufgebaut wird und sich noch in einem inkonsistenten Zustand befindet.



FIX/Win befindet sich im *Edit-Bitmap*-Modus. Ein Bitmap kann zum Editieren angeklickt werden.



FIX/Win befindet sich im *Zeicheninfo*-Modus. Es kann ein Zeichen angeklickt werden, zu dem Informationen ausgegeben werden.

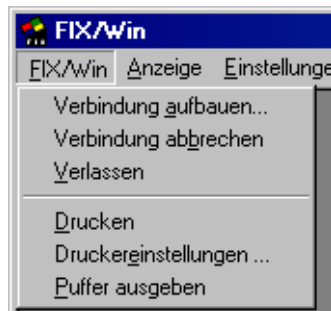
3 Aufbau und Funktion des Hauptmenüs

Zur Bedienung von FIX/Win selbst dient das Windows-Menü des Programms. Es wird weitgehend analog jedem anderen Windows-Menü bedient. Einziger Unterschied ist, dass die Taste F10 nicht mehr zum Start des Menüs verwendet werden kann, sondern FIX-Anwendungen vorbehalten ist. Es folgt eine Beschreibung der einzelnen Menüpunkte.

3.1 Hauptmenüpunkt "FIX/Win"

Mit diesem Menü kann die Verbindung zur FIX-Anwendung aufgebaut und abgebrochen werden. Außerdem kann das Programm verlassen werden.

Abb. 3 Hauptmenüpunkt "FIX/Win"



Menüpunkt "Verbindung aufbauen"

Durch Anwahl dieses Menüpunktes erscheint die Dialogbox zum Verbindungsaufbau (vgl. [Dialog zum Verbindungsaufbau](#) auf Seite 27). Hier kann unter den verschiedenen Anwendungen, die in der Programmtabelle eingetragen sind, ausgewählt werden. Nach Eingabe von Benutzer und Passwort kann dann die Verbindung zum FIX-Programm auf dem Hostrechner durch Anklicken des Buttons "Verbinden" aufgebaut werden. Durch Betätigung des Buttons "Abbruch" werden alle Eingaben verworfen und die Dialogbox entfernt.¹ Das Feld zur Eingabe des Benutzernamens wird vorbelegt aus (siehe [/user <User>](#) auf Seite 19):

- Angabe zum Kommandozeilenparameter "/user", oder falls der Schalter nicht gesetzt war:
- Inhalt der Umgebungsvariable USER, oder falls Variable nicht gesetzt war:
- Inhalt der Umgebungsvariable LOGNAME

Menüpunkt "Verbindung abbrechen"

Über diesen Menüpunkt kann eine laufende FIX-Anwendung zwingend abgebrochen werden, indem die Verbindung geschlossen wird. Ein solcher Abbruch beendet die Anwendung jedoch nicht korrekt, was zu korrupten Daten führen kann. Deshalb ist es ratsam, für einen solchen Fall entsprechende Vorkehrungen in der Applikation zu treffen.

Da es nur dann sinnvoll ist, eine Verbindung abzubauen, wenn vorher eine aufgebaut wurde, ist der Menüpunkt nur in diesem Fall aktiv. Genauso sind einige andere Menüpunkte nur aktiv, wenn eine Verbindung zu einer FIX-Anwendung besteht.

Menüpunkt "Verlassen"

Hiermit kann FIX/Win selbst beendet werden. Dieser Menüpunkt ist nur aktiv, wenn keine Verbindung zu einer FIX-Anwendung besteht. Das heißt, dass zuerst die FIX-Anwendung beendet werden muss, bevor FIX/Win beendet werden kann.

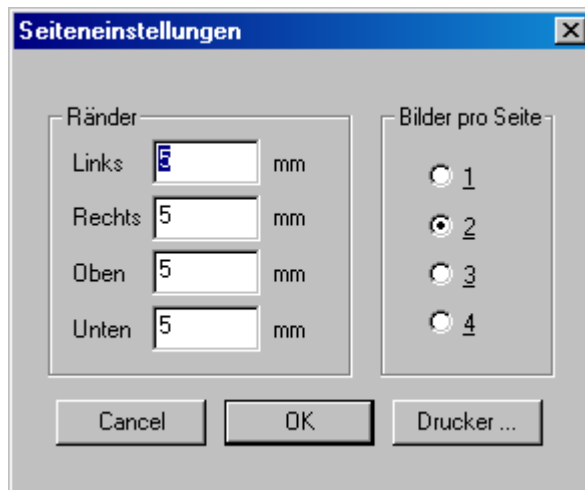
1. Zur grundsätzlichen Bedienung von Dialogboxen wird auf das Windows-Handbuch verwiesen.

Menüpunkt "Drucken"

Über diesen Menüpunkt wird der Inhalt des aktuellen FIX/Win-Fensters als Hardcopy auf den Drucker ausgegeben. Der Drucker sowie weitere Optionen können über den folgenden Menüpunkt eingestellt werden.

Menüpunkt "Druckereinstellungen ..."

Über diesen Menüpunkt kann der folgende Dialog gestartet werden:



In den Eingabefeldern kann ein Randbereich für den Ausdruck definiert werden. Die Angabe Bilder pro Seite definiert, wieviele Bilder auf einer Seite ausgegeben werden. Erst wenn diese Anzahl Hardcopies erreicht wird, dann wird die Seite auf den Drucker ausgegeben. Der Drucker kann über den Button "Drucker ..." ausgewählt und eingestellt werden. Dazu wird der windowstypische Dialog zur Druckereinrichtung eingeblendet.

Menüpunkt "Puffer ausgeben"

Wenn die eingestellte Anzahl Hardcopies nicht erreicht wurde, dann kann über diesen Menüpunkt die Ausgabe der Seite erzwungen werden.

3.2 Hauptmenüpunkt "Anzeige"

Abb. 4 Hauptmenüpunkt "Anzeige"



Menüpunkt "Meldungsfenster"

Alle Fehlermeldungen und Warnungen werden in einem *Meldungsfenster* ausgegeben.

Mit dem Untermenüpunkt *"anzeigen"* kann das Meldungsfenster ein- und ausgeblendet werden. Tritt eine neue Fehlermeldung auf, wird es automatisch eingeblendet. Beim Einblenden eines frei verschiebbaren Fensters wird es am unteren Rand des Hauptfensters eingeblendet.

Mit dem Untermenüpunkt *"löschen"* kann der Inhalt des Meldungsfensters gelöscht werden. So kann z.B. beim Eintreffen neuer Meldungen festgestellt werden, welche Meldungen seit dem letzten Löschen neu hinzugekommen sind.

Mit dem Untermenüpunkt *"als Fenster"* kann zwischen der Darstellung in einem eigenen Fenster und der Darstellung als Teilfenster des Hauptfensters umgeschaltet werden. Ein eigenes Fenster kann frei in der Größe verändert werden und verschoben werden. Bei einem Fenster, das Teil des Hauptfensters ist, kann nur die Höhe des Fensters durch Ziehen an der Linie über der Titelzeile geändert werden. Die anderen Ausmaße des Fensters passen sich dem Hauptfenster an.

Mit dem Untermenüpunkt *"Inhalt speichern"* kann der Inhalt des Meldungsfensters in einer Datei abgelegt werden.

Menüpunkt *"IconBox"*

An der IconBox können verschiedene Einstellungen vorgenommen werden, die alle über diesen Menüpunkt gesteuert werden. Hierzu wird ein Untermenü zur Aktivierung oder Deaktivierung der verschiedenen Einstellungen mit folgenden Optionen aufgerufen.:

- *anzeigen* Die IconBox wird ausgeblendet wenn sie vorher sichtbar war bzw. angezeigt wenn sie vorher unsichtbar war.
- *horizontal* Hiermit wird entschieden, ob die Icons nebeneinander oder untereinander dargestellt werden. Wird dieser Menüpunkt ausgewählt, während die IconBox ausgeblendet ist, so wird die IconBox zur Anzeige gebracht.

Menüpunkt *"Statuszeile"*

Mit diesem Menüpunkt kann die Statuszeile am unteren Rand des Fensters von FIX/Win ein- und ausgeblendet werden. Das Ausblenden dieser Zeile kann sinnvoll sein, um mehr Platz für die FIX-Anwendung im Fenster zu schaffen. Ist die Statuszeile eingeblendet, werden dort folgende Informationen angezeigt:

- der Text "Keine Verbindung", wenn keine Verbindung besteht
- Name des Programms, Benutzers und Hosts, wenn eine Verbindung besteht
- ein Signal, das zwischen Rot und Grün wechselt und anzeigt, wann Daten über das Netz gesendet werden. Dabei bedeutet Rot, dass gerade Daten empfangen werden.

Menüpunkt *"Tooltips"*

Über diesen Menüpunkt können Tooltips auf Paintareas und Feldern global an- und ausgeschaltet werden.

Menüpunkt *"Bild neu aufbauen"*

Sollte aus irgendeinem Grund der Bildschirminhalt zerstört werden, sei es auf Seite von Windows oder auf Seite der FIX-Anwendung, kann er mit diesem Menüpunkt neu aufgebaut werden. Hierzu wird der FIX-Anwendung das Event *gl* gesendet, woraufhin ein FIX-Programm im Standardfall den Bildschirm restauriert¹.

1. Die Eventbehandlung der FIX-Anwendung kann jedoch dieses Event auch für eigene Zwecke nutzen, so dass in diesem Fall ein Bildneuaufbau nicht möglich ist.

Menüpunkt "Standardgröße"

Bei der Auswahl dieses Menüpunktes wird das aktuelle Container-Fenster auf die Größe gebracht, bei der eine Darstellung von Scrollbars nicht notwendig ist. Bei der Verwendung des Schalters */single* wird zusätzlich das Hauptfenster dem Container-Fenster angepasst.

Menüpunkt "Kopieren"

Die Untermenüpunkte dieses Menüpunktes dienen zum Kopieren von Zeichen aus dem FIX/Win-Fenster.

Untermenüpunkt "Kopieren/abschalten"

Mittels dieses Menüpunktes werden eingeschaltete Markierungsmodi abgeschaltet.

Untermenüpunkt "Kopieren/Bereich"

Der Mauscursor wird umgeschaltet, wodurch angedeutet wird, dass FIX/Win sich im Bereichsmarkierungsmodus befindet. Nun kann mit der Maus ein Startpunkt angeklickt werden und von dort durch Ziehen ein Bereich markiert werden. Dieser Bereich wird nach dem Loslassen des Mausbuttons in die Zwischenablage kopiert.

Abb. 5 Mauszeiger im Bereichsmarkierungsmodus



Untermenüpunkt "Kopieren/Feld"

Mit diesem Markierungsmodus kann man ein Feld innerhalb der Maske anklicken, welches in die Zwischenablage kopiert wird.

Abb. 6 Mauszeiger im Feldmarkierungsmodus



Untermenüpunkt "Kopieren/Objekt"

Mit diesem Markierungsmodus kann durch Anklicken ein ganzes Objekt (eine Maske, ein Menü, ein Selo, eine Choice ...) markiert und in die Zwischenablage kopiert werden.

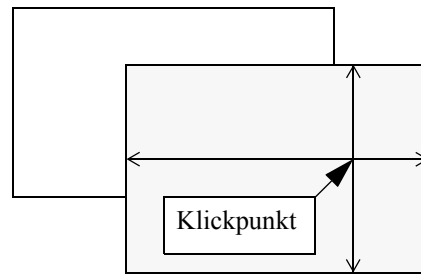
Abb. 7 Mauszeiger im Objektmarkierungsmodus



Das Anklicken von Objekten, die von anderen teilweise verdeckt werden, ist zwar möglich, aber mit Problemen behaftet. Der Grund liegt zum einen im verwendeten Algorithmus, der die Fläche eines Objektes als Rechteck zu ermitteln versucht, und zum anderen in der Tatsache, dass oben liegende Objekte den Bildschirmspeicher der darunter liegenden Objekte überschreiben. Zum besseren Verständnis werden im folgenden verschiedene Möglichkeiten des Anklickens aufgezeigt. Dabei wird folgendes Verfahren zur Ermittlung der Rechteckfläche zu Grunde gelegt:

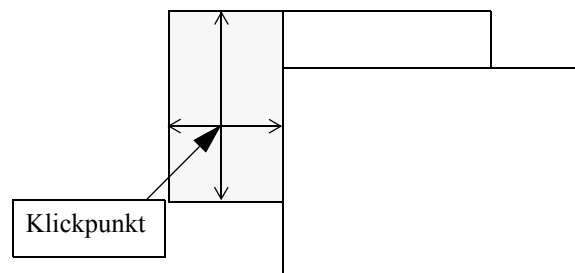
Von dem angeklickten Punkt wird in alle 4 Richtungen solange nach außen gegangen, bis ein anderes Objekt oder der Bildrand erreicht wird. Die so gefundenen Werte dienen als obere, untere, linke und rechte Kante des Rechtecks, das markiert wird.

Die folgende Grafik zeigt das Markieren eines oben liegenden Objektes:



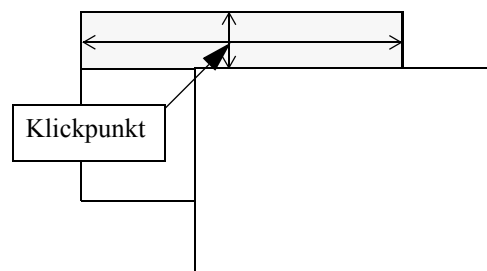
Alle Kanten des Objekts werden korrekt ermittelt und der Inhalt kann in die Zwischenablage kopiert werden. Anders, wenn das verdeckte Objekt angeklickt wird. Hier sind 3 Fälle möglich:

- Anklicken neben dem Objekt, welches das Objekt verdeckt:



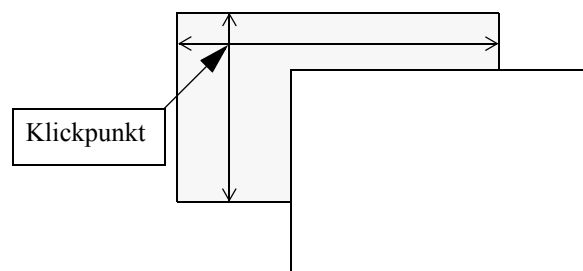
Es wird nur das Rechteck markiert, welches neben dem verdeckenden Objekt liegt.

- Anklicken über dem verdeckenden Objekt:



Hier wird nur das Rechteck über dem verdeckenden Objekt markiert und in die Zwischenablage kopiert.

- Anklicken in diagonalem Abstand von einer Ecke des verdeckenden Objekts:



Hier werden alle Kanten des angeklickten Objektes korrekt gefunden und markiert. Allerdings beim Kopieren in die Zwischenablage wird der verdeckte Teil nicht aus dem angeklickten Objekt gelesen, sondern aus dem verdeckenden Objekt, da dieses ja den Bildschirmspeicher an dieser Stelle überschreibt.

Untermenüpunkt "Kopieren/im Rahmen"

Oft besitzt eine Maske mehrere Submasken. Dieser Markierungsmodus dient dazu eine Hauptmaske mit allen Submasken zu markieren.

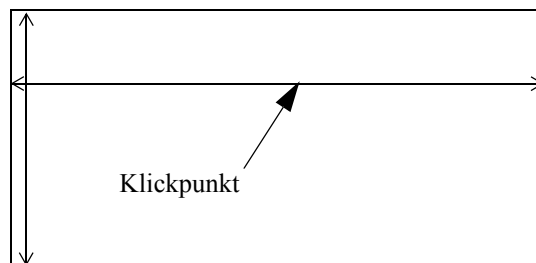
Abb. 8 Mauszeiger im Rahmenmarkierungsmodus

Es wird davon ausgegangen, dass die Hauptmaske, sowie die Submasken von einem Rahmen umgeben sind. Dieser Rahmen muss folgende Bedingungen erfüllen:

- Er muss aus FIX Semigrafikzeichen für *schattierte Linien* bestehen (siehe Kapitel über Semigrafik im FIX Applikationsgenerator Handbuch)
- Der Rahmen darf nur an der oberen und unteren Linie unterbrochen sein, und dies nicht an der Position hinter der linken oberen und der linken unteren Ecke.
- Innerhalb des Rahmens dürfen keine *schattierten Linien* verwendet werden.

Nur wenn diese Bedingungen erfüllt sind, arbeitet das im folgende beschriebene Verfahren korrekt:

Die linke und rechte Kante des markierten Rechtecks werden ermittelt, indem vom Klickpunkt aus so lange nach links und rechts gewandert wird, bis eine schattierte Linie gefunden wird. Die obere und untere Kante werden ermittelt, indem eine Position hinter der linken Kante so weit nach oben und unten gewandert wird bis eine schattierte Linie gefunden wird. Die folgende Grafik veranschaulicht das Verfahren:



Auch hier ist Vorsicht beim Anklicken von verdeckten Objekten geboten.

3.3 Hauptmenüpunkt "Einstellungen"

Mit diesem Menü können verschiedene Eigenschaften von FIX/Win eingestellt. Sämtliche Einstellungen werden als Ressourcen abgelegt und beim Start der FIX-Anwendung geladen.

Abb. 9 Hauptmenüpunkt "Einstellungen"

Menüpunkt "Bildgröße"

Hiermit wird die Größe des FIX/Win-Fensters eingestellt. Es sind bis zu drei Einstellungen möglich. Jeder Einstellung ist eine bestimmte Zellengröße zugeordnet. Die Größe des Fensters auf dem Bildschirm ist von dieser zellengröße und von der Anzahl der Zeilen und Spalten, die die FIX-Anwendung benötigt, abhängig. Die Bezeichnungen und die An-

zahl der Menüpunkte zur Einstellung der Größe können über Ressourcen definiert werden. Daher ist das Untermenü von der Anwendung abhängig.

Menüpunkt "Stil"

Mit diesem Menüpunkt können verschiedene Bildstile zur Darstellung des FIX/Win-Fensters eingestellt werden. Diese Stile definieren die Farben und das Aussehen der Rahmen und Linien des FIX-Programms. Die Bezeichnungen und die Anzahl der Menüpunkte zur Einstellung des Bildstils können über Ressourcen definiert werden. Daher ist das Untermenü von der Anwendung abhängig.

Menüpunkt "Kopiermodus ..."

Dieser Menüpunkt dient zur Einstellung des Verhaltens im Kopiermodus. Er öffnet eine Dialogbox mit der folgende Optionen eingestellt werden können:

- Option "Kopieren starten":
Wird diese Option auf "automatisch" eingestellt, dann wird nach dem Markieren eines Bereichs des Ausgabebildschirms dieser direkt in die Zwischenablage kopiert. Wird die Option auf "nach Abfrage" eingestellt, dann wird vor dem Kopieren abgefragt, welche Teile des markieren Bereichs in die Zwischenablage kopiert werden sollen. Diese Abfrage entspricht der Option "Kopieren von". Weiterhin kann innerhalb dieser Abfrage der Kopiervorgang abgebrochen werden, um zu verhindern, dass der aktuelle Inhalt der Zwischenablage überschrieben wird.
- Option "Kopieren von":
Steht die Option "Kopieren starten" auf "automatisch", kann mit dieser Option bestimmt werden, welche Teile des markierten Bereichs in die Zwischenablage eingefügt werden sollen. "Felder" kopiert nur die Feldinhalte als Text (Grafikfelder können also nicht kopiert werden). Alle anderen Zeichen werden durch Leerzeichen ersetzt. "Alles" kopiert den gesamten markieren Bereich.
- Option "Rechte Maustaste"
Mit dieser Option kann die rechte Maustaste mit einem Markierungsmodus belegt werden. "Feld markieren" stellt die rechte Maustaste so ein, dass beim Anklicken eines Feldes dieses markiert wird. Analog arbeiten die Werte "Objekt markieren" und "Rahmen markieren". Wird die Option jedoch auf "Taste senden" eingestellt, dann wird beim Klicken der rechten Maustaste das in der Tastenbelegungsdatei hinterlegte FIX Event bzw. das entsprechende BT_...-Event an die Anwendung gesendet. Weiterhin ist zu bemerken, dass nur in dieser Einstellung die mittlere Maustaste durch Klicken der rechten und linken emuliert werden kann.
- Option "Einschalten über Timeout"
Ist diese Option eingeschaltet, dann kann der Markierungsmodus "Kopieren/Bereich" (siehe Beschreibung des Menüpunktes Ansicht/Kopieren) durch Festhalten der linken Maustaste eingeschaltet werden. Wird die Maustaste eine längere Zeit (ca 0.5 sec) festgehalten, ändert sich der Mauscursor in ein Markierungssymbol, mit welchem durch Ziehen ein Bereich markiert und in die Zwischenablage kopiert werden kann. Schaltet man diese Option ab, wird dieses Verhalten unterdrückt. Somit kann verhindert werden, dass die Zwischenablage versehentlich durch zu langes Festhalten der Maustaste überschrieben wird.

Menüpunkt "Speichern"

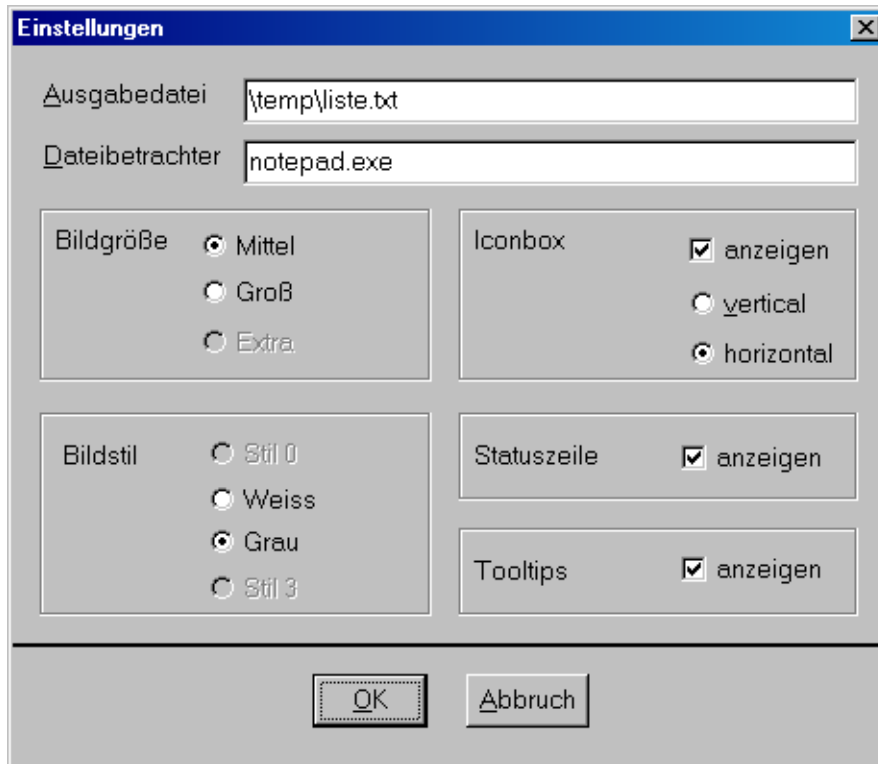
Durch Anwahl dieses Menüpunktes werden alle Einstellungen als Ressourcen abgelegt. Dabei werden folgende Werte abgespeichert:

- aktueller Bildstil
- aktuelle Bildgröße
- Anzeige der Statuszeile
- Anzeige der IconBox
- IconBox vertikal - horizontal

- Name der Ausgabedatei
- Name des Dateibetrachters
- Fensterposition des FIX/Win-Fensters

Menüpunkt "Übersicht"

Abb. 10 Dialogbox, von Menüpunkt "Übersicht" gestartet



Mit diesem Menüpunkt sind alle Optionen übersichtlich in einer Dialogbox einstellbar. Hierzu gehören unter anderem die auch getrennt über Menüpunkte einstellbaren Optionen für IconBox, Statuszeile und Bildgröße/Stil. Die Einstellmöglichkeiten von Bildgröße und Bildstil sind wie in den entsprechenden Menüs von der Anwendung abhängig und über Ressourcen einstellbar. Im Feld "Ausgabedatei" muss eine Datei angegeben werden, in die die Textausgabe des FIX-Programms geschrieben wird und im Feld "Dateibetrachter" ein Windows-Programm, das diese Datei anzeigt (siehe [Ausgabe von Listen](#) auf Seite 39). Durch Anklicken von "OK" wird die Dialogbox beendet, und die Einstellungen werden aktiviert. Mit dem Anklicken von "Abbruch" wird die Dialogbox abgebrochen.

3.4 Hauptmenüpunkt "Hilfe"

Dieser Menüpunkt besitzt zwei Untermenüpunkte. Mit "Info" können eine Copyright-Meldung und Versionsinformationen angezeigt werden. Mit dem Untermenüpunkt "anzeigen" wird das FIX/Win Online Handbuch aufgerufen.

4 Ausgabe von Listen

In FIX-Anwendungen besteht häufig die Möglichkeit, Ausgabelisten zu generieren. In den meisten Fällen werden diese auf dem Drucker ausgegeben, der vom Applikationsserver aus angesteuert wird. Es ist jedoch auch möglich, eine Ausgabeliste auf dem Bildschirm anzuzeigen. Um diese Ausgabe zu FIX/Win zu leiten, sollte die FIX-Anwendung ein Programm aufrufen, das direkt auf die Standardausgabe schreibt. Dieses Programm darf keine interaktiven Abfragen beinhalten, da diese von FIX/Win nicht bearbeitet werden können. Alle Daten, die auf die Standardausgabe geschrieben werden, legt FIX/Win in einer Datei auf dem Frontend-Host ab. Wenn die Ausgabe abgeschlossen ist, wird von FIX/Win ein Betrachter zum Anzeigen der Datei aufgerufen. Die Datei, in die die Ausgabe auf dem Frontend-Host geschrieben wird, kann in der Dialogbox *Einstellungen* (siehe [Hauptmenüpunkt "Einstellungen"](#) auf Seite 36) vorgegeben werden. Ebenso muss dort ein Programm zum Betrachten der Datei festgelegt werden.

Damit FIX/Win erkennt, wann eine Ausgabe gestartet bzw. beendet wird, sollte zum Start des externen Programmes eine der folgenden Möglichkeiten verwendet werden:

- Verwendung der Funktion `execute_cmd()` aus der FIX-Bibliothek
- Angabe der Aktion `sh <Programm>` bei einem Menü oder einer Maske
- Verwendung der Funktion `exec_ms_command()` aus der FIX-Bibliothek

3 Anpassung von FIX/Win

FIX/Win arbeitet ohne Anpassungen direkt mit Anwendungen zusammen, die mit FIX ab Version 3.1.0 erstellt wurden. Soll das Verhalten und das optische Erscheinungsbild von FIX/Win geändert werden, so kann dies durch Veränderung der Konfigurationsdateien erreicht werden. Folgende Punkte können dabei beeinflusst werden:

- Aufbau der IconBox in Bezug auf Anzahl der Icons, Eventbelegung, Aussehen und Reihenfolge
- Belegung der Tastatur mit FIX-Events
- Zuordnung von Farben zu FIX-Elementen für jeden Bildschirmstil
- Hinzufügen oder Verändern von Semigrafikzeichen für jeden Bildschirmstil

Grundsätzlich sind alle Konfigurationsdateien im Arbeitsverzeichnis von FIX/Win als Standardkonfiguration enthalten. Zur Änderung einer Konfiguration ist die Standarddatei als Vorlage in das Gruppenverzeichnis zu kopieren und dort den Erfordernissen anzupassen.

1 Kodierung der Konfigurationsdateien

Beim Einlesen der textuellen Konfigurationsdateien werden alle Zeichen von FIX/Win grundsätzlich in 16-Bit breite Zeichen zur internen Verwendung umgewandelt. Der Zeichensatz der Datei wird dabei, wie unter Windows üblich, durch die *Byte Order Mark* bestimmt. Hierbei handelt es sich um zwei bis drei Bytes, die am Anfang der Datei stehen und somit die Kodierung der Datei definieren. Damit sind die Kodierungen *ANSI*, *UTF-8*, *Unicode* und *Unicode Big Endian* möglich. Als Editor zum Bearbeiten und zum Konvertieren in einen anderen Zeichensatz muss ein Editor verwendet werden, der die *Byte Order Mark* unterstützt (zB. Notepad). Für Dateien, die keine *Byte Order Mark* besitzen, wird der Zeichensatz *ANSI* angenommen. Konfigurationsdateien von älteren Versionen können deshalb - zumindest, was den Zeichensatz betrifft - für FIX/Win 3.1.0 übernommen werden.

Ausnahme von dieser Regel bildet die Datei `stdicon.fix`. Für diese Datei darf die Kodierung *Unicode* oder *Unicode Big Endian* nicht verwendet werden.

Beim Schreiben von Dateien verwendet FIX/Win immer die Kodierung *UTF-8* und erzeugt auch die dazugehörige *Byte Order Mark*.

2 Aufbau eines Gruppenverzeichnisses

Das Gruppenverzeichnis enthält Dateien, die als Ersatz für Dateien von FIX/Win verwendet werden. Beim Laden einer Datei prüft FIX/Win immer erst, ob die entsprechende Datei im Gruppenverzeichnis vorhanden ist. Darüber hinaus ist es möglich und üblich im Gruppenverzeichnis weitere Dateien zu hinterlegen, die für eine bestimmte FIX-Anwendung oder eine Gruppe von FIX-Anwendungen notwendig ist. So wird dort beispielsweise oft die Benutzerbibliothek installiert.

Die folgenden Abschnitte beschreiben den Aufbau der Unterverzeichnisse eines Gruppenverzeichnisses.

config

Hier liegen alle Konfigurationsdateien die nicht das Look&Feel direkt betreffen:

- `stdicon.fix`: Die IconBox-Definitionsdatei.
- `stdkey.fix`: Die Tasten-Definitionsdatei.
- `*.frc`: FIX/Win-Ressourcendateien mit anwendungsabhängigen und anwendungsunabhängigen Ressourcen.

LookAndFeel

Dieses Verzeichnis kann für jeden Stil ein Unterverzeichnis mit Dateien zum Look&Feel des FIX/Win-Fensters und ein Unterverzeichnis mit Dateien zum Look&Feel der IconBox enthalten:

- `Fixwin-LAF-0`: Dateien zum Look&Feel im Stil 0
- `Fixwin-LAF-1`: Dateien zum Look&Feel im Stil 1
- `Fixwin-LAF-2`: Dateien zum Look&Feel im Stil 2
- `Fixwin-LAF-3`: Dateien zum Look&Feel im Stil 3
- `IconBox-LAF-0`: Dateien zum IconBox-Look&Feel im Stil 0
- `IconBox-LAF-1`: Dateien zum IconBox-Look&Feel im Stil 1
- `IconBox-LAF-2`: Dateien zum IconBox-Look&Feel im Stil 2
- `IconBox-LAF-3`: Dateien zum IconBox-Look&Feel im Stil 3

Zusätzlich zu den *stilabhängigen* Verzeichnissen können die *stilunabhängigen* Verzeichnisse

- `Fixwin-LAF`: Dateien zum Look&Feel
- `IconBox-LAF`: Dateien zum IconBox-Look&Feel

vorhanden sein. Die Dateien aus diesen beiden Verzeichnissen werden dann verwendet, wenn zu einem Stil kein stilabhängiges Unterverzeichnis vorhanden ist oder eine Datei in dem stilabhängigen Verzeichnis nicht vorhanden ist. Sie definieren somit den Default des Gruppenverzeichnisses.

Fixwin-LAF

Das Verzeichnis `Fixwin-LAF` ist das Defaultverzeichnis für die stilunabhängigen Dateien eines Look&Feels. Die hier beschriebenen Dateien können in allen anderen stilabhängigen Verzeichnissen (`Fixwin-LAF-<NR>`) ebenso vorhanden sein:

- `FW-M-L15.FNT`: Font für die Größe Medium und den Zeichensatz Windows-1252.
- `FW-L-L15.FNT`: Font für die Größe Large und den Zeichensatz Windows-1252.
- `FW-H-L15.FNT`: Font für die Größe Huge und den Zeichensatz Windows-1252.
- `FW-M-L2.FNT`: Font für die Größe Medium und den Zeichensatz Windows-1250.
- `FW-L-L2.FNT`: Font für die Größe Large und den Zeichensatz Windows-1250.
- `FW-H-L2.FNT`: Font für die Größe Huge und den Zeichensatz Windows-1250.
- `caret-M.bmp`: Bitmap zur Darstellung des Carets in der Größe Medium.
- `caret-L.bmp`: Bitmap zur Darstellung des Carets in der Größe Large.
- `caret-H.bmp`: Bitmap zur Darstellung des Carets in der Größe Huge.
- `KeyFrmActive-M.bmp`: Bitmap zur Darstellung des Rahmens eines Tasten-Labels in aktivem Zustand in der Größe Medium.
- `KeyFrmActive-L.bmp`: Bitmap zur Darstellung des Rahmens eines Tasten-Labels in aktivem Zustand in der Größe Large.
- `KeyFrmActive-H.bmp`: Bitmap zur Darstellung des Rahmens eines Tasten-Labels in aktivem Zustand in der Größe Huge.

- `KeyFrmInActive-M.bmp`: Bitmap zur Darstellung des Rahmens eines Tasten-Labels in inaktivem Zustand in der Größe Medium.
- `KeyFrmInActive-L.bmp`: Bitmap zur Darstellung des Rahmens eines Tasten-Labels in inaktivem Zustand in der Größe Large.
- `KeyFrmInActive-H.bmp`: Bitmap zur Darstellung des Rahmens eines Tasten-Labels in inaktivem Zustand in der Größe Huge.
- `coltab.fix`: Datei mit Farbdefinitionen der einzelnen FIX-Elemente.
- `bitmap-M.bmp/.bif`: gepackte Bitmaps für Semigrafikzeichen in der Größe Medium mit entsprechender Strukturdatei (`*.bif`)
- `bitmap-L.bmp/.bif`: gepackte Bitmaps für Semigrafikzeichen in der Größe Large mit entsprechender Strukturdatei (`*.bif`)
- `bitmap-H.bmp/.bif`: gepackte Bitmaps für Semigrafikzeichen in der Größe Huge mit entsprechender Strukturdatei (`*.bif`)
- `keyBitmap-M.bmp/.bif`: gepackte Bitmaps zur Tastendarstellung in der Größe Medium mit entsprechender Strukturdatei (`*.bif`)
- `keyBitmap-L.bmp/.bif`: gepackte Bitmaps zur Tastendarstellung in der Größe Large mit entsprechender Strukturdatei (`*.bif`)
- `keyBitmap-H.bmp/.bif`: gepackte Bitmaps zur Tastendarstellung in der Größe Huge mit entsprechender Strukturdatei (`*.bif`)
- `lookAndFeel.frc`: FIX/Win-Ressourcdatei mit Einstellungen zum LookAndFeel

IconBox-LAF

Das Verzeichnis `IconBox-LAF` ist das Defaultverzeichnis für die stilabhängigen Dateien eines `IconBox-Look&Feels`. Die hier beschriebenen Dateien können in allen anderen stilabhängigen Verzeichnissen (`IconBox-LAF-<NR>`) ebenso vorhanden sein:

- `iconbox.frc`: FIX/Win-Ressourcdatei der `IconBox`.
- `frmActive.bmp`: Bitmap zur Darstellung eines aktiven Rahmens für ein Icon.
- `frmInActive.bmp`: Bitmap zur Darstellung eines inaktiven Rahmens für ein Icon.
- `frmDisabled.bmp`: Bitmap zur Darstellung eines deaktivierten Rahmens für ein Icon.
- `frmActive97.bmp`: Bitmap zur Darstellung eines aktiven Rahmens im 97-Stil für ein Icon.
- `frmActive97.bmp`: Bitmap zur Darstellung eines inaktiven Rahmens im 97-Stil für ein Icon.
- `frmSelect97.bmp`: Bitmap zur Darstellung eines selektierten Rahmens im 97-Stil für ein Icon.
- `*.bmp`: Bitmaps zur Darstellung der Icons.

LookAndFeel-src

Das Verzeichnis `LookAndFeel-src` enthält die ungepackten Bitmaps. Es hat die gleiche Struktur wie das Verzeichnis `LookAndFeel`. Statt der gepackten Bitmapdateien (`bitmap-[M-H].bmp/.bif` und `keyBitmap-[M-H].bmp/.bif`) sind jedoch Unterverzeichnisse mit dem gleichen Basisnamen vorhanden, die die Bitmaps enthalten, die gepackt werden. Das Programm `bmpack` liest die Dateien aus `LookAndFeel-src` und legt die gepackten Ergebnisse im Verzeichnis `LookAndFeel` ab.

2.1 Verfahren zum Einlesen von Dateien

Dateien aus dem Unterverzeichnis `config`

Bei der Tastendefinitionsdatei und der Icondefinitionsdatei wird als erstes versucht, die entsprechende Datei aus dem Unterverzeichnis `config` des Gruppenverzeichnisses zu lesen. Wenn die Datei dort nicht vorhanden ist, dann wird die Datei aus dem Unterverzeichnis `config` des Arbeitsverzeichnisses von FIX/Win verwendet.

Für anwendungsabhängige und anwendungsunabhängige FIX/Win-Ressourcendateien (`*frc`) wird ein etwas abgewandtes Verfahren benutzt, das im Abschnitt *Ressourcendateien von FIX/Win* auf Seite 80 beschrieben wird.

Dateien aus den Unterverzeichnissen `Fixwin-LAF` und `IconBox-LAF`

Der Zugriff auf diese Dateien hängt vom eingestellten Bildschirm-Stil ab. Jeder der Stile besitzt eine Nummer aus dem Bereich 0-3. Die Nummer bildet den Präfix eines Verzeichnisses. `IconBox-LAF-2` enthält beispielsweise die Dateien zum Look&Feel im Stil 2. Als erstes wird versucht die zu einem Stil passende Datei aus dem entsprechenden Unterverzeichnis des Gruppenverzeichnisses zu laden. Ist sie dort nicht vorhanden (oder existiert das Verzeichnis nicht), dann wird versucht sie aus dem stilunabhängigen Verzeichnis (das Verzeichnis ohne die Stilnummer als Präfix) des Gruppenverzeichnisses zu laden. Ist sie auch dort nicht vorhanden, dann wird das stilabhängige Verzeichnis des FIX/Win Arbeitsverzeichnisses verwendet. Wenn die Datei dort auch nicht vorhanden ist, dann wird sie aus dem stilunabhängigen Verzeichnis des FIX/Win Arbeitsverzeichnisses geladen.

Beispiel

Wenn die Größe 2 ("Huge") eingestellt wurde, dann werden die Bitmaps für Semigrafikzeichen aus der Datei `bitmap-H.bmp` gelesen. Für den Bildstil 3 ergibt sich folgende Suchreihenfolge:

```
c:/grpdir/LookAndFeel/Fixwin-LAF-3/bitmap-H.bmp
c:/grpdir/LookAndFeel/Fixwin-LAF/bitmap-H.bmp
c:/fixwin/LookAndFeel/Fixwin-LAF-3/bitmap-H.bmp
c:/fixwin/LookAndFeel/Fixwin-LAF/bitmap-H.bmp
```

`c:/grpdir` ist dabei das Gruppenverzeichnis und `c:/fixwin` das Arbeitsverzeichnis von FIX/Win.

Durch die Verwendung von stilunabhängigen Verzeichnissen ist es möglich, Dateien zu definieren, die in mehreren Stilen verwendet werden. So sind beispielsweise die Fonts für den Zeichensatz ISO-8859-15 für alle Stile außer dem Stil 1 gleich. Die Font-Dateien aller anderen Stile liegen deshalb im Unterverzeichnis `Fixwin-LAF` und die von Stil 1 im Unterverzeichnis `Fixwin-LAF-1` des Arbeitsverzeichnisses von FIX/Win.

Das hier beschriebene Verfahren wird von FIX/Win für alle Dateien verwendet, die unterhalb des Verzeichnisses `LookAndFeel` liegen. Das Programm `bmpack` zum Packen von Bitmaps verwendet ein ähnliches Verfahren. Das Programm erzeugt die Dateien `bitmap-[HML].bmp/.bif` und `keyBitmap-[HML].bmp/.bif` indem es die Bitmaps aus den Unterverzeichnissen von `LookAndFeel-src` lädt und zusammen packt. Dabei werden mehrere Verzeichnisse nacheinander durchlaufen. Aus dem ersten Verzeichnis werden alle Bitmaps geladen. Aus den weiteren Verzeichnissen werden nur noch die Bitmaps geladen, die in den vorherigen Verzeichnissen nicht vorhanden waren und so noch nicht geladen sind. Wenn alle Verzeichnisse für eine Stil-Größenkombination durchlaufen wurden, wird das gepackte Bitmap erzeugt und in dem entsprechenden Unterverzeichnis von `LookAndFeel` abgelegt. Beim Laden werden die Verzeichnisse in folgender Reihenfolge durchlaufen:

- stilabhängiges Verzeichnis des Gruppenverzeichnisses
- stilunabhängiges Verzeichnis des Gruppenverzeichnisses
- stilabhängiges Verzeichnis des Arbeitsverzeichnisses von FIX/Win
- stilunabhängiges Verzeichnis des Arbeitsverzeichnisses von FIX/Win

Beispiel:

Für die Größe 2 ("Huge") und den Bildstil 3 erzeugt `bmpack` die Dateien `c:/grpdir/LookAndFeel/Fixwin-LAF-3/bitmap-H.bmp` und `c:/grpdir/LookAndFeel/Fixwin-LAF-3/bitmap-H.bif`. Die Bitmaps dafür werden aus folgenden Unterverzeichnissen in der angegebenen Reihenfolge gelesen:

```
c:/grpdir/LookAndFeel-src/Fixwin-LAF-3/bitmap-H
c:/grpdir/LookAndFeel-src/Fixwin-LAF/bitmap-H
c:/fixwin/LookAndFeel-src/Fixwin-LAF-3/bitmap-H
c:/fixwin/LookAndFeel-src/Fixwin-LAF/bitmap-H
```

3 Konfiguration der IconBox

Die IconBox enthält mehrere Icons, die beim Anklicken ein Event oder mehrere Events an die FIX-Anwendung senden. Die Konfiguration der IconBox unterteilt sich in zwei unterschiedliche Bereiche:

- *Der grundsätzliche Aufbau der IconBox.* Hierzu zählt die Menge der Icons, die Reihenfolge, die Events, die beim Anklicken gesendet werden und die Information, welche Icons zu einem bestimmten Zeitpunkt anklickbar sind.
- *Das Aussehen der IconBox.* Hierzu zählen die verwendeten Bitmaps für die Icons, die Größe der Icons, Farbeinstellungen für Hintergrund und Transparenz, Abstände der Icons zum Rand und die Bitmaps für die Rahmen der Icons.

3.1 Der grundsätzliche Aufbau der IconBox

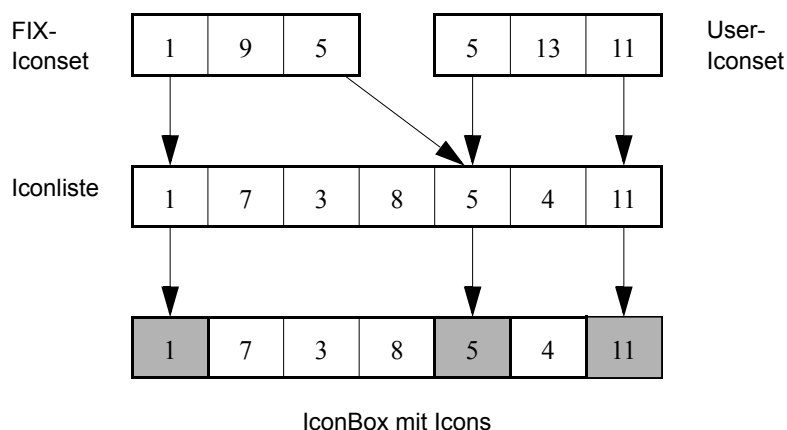
Der grundsätzliche Aufbau wird durch die Datei

`config/stdicon.fix`

definiert. Sie wird als Icondefinitionsdatei bezeichnet. FIX/Win verwendet zur Definition der Icons das Konzept von *Iconset* und *Iconliste*. Eine *Iconliste* definiert alle Icons der IconBox, deren Reihenfolge und die Zwischenräume. Es können mehrere Iconlisten zur Auswahl definiert werden. Welche Iconliste angezeigt wird, kann von der FIX-Anwendung bestimmt werden. Je nach Situation kann zur Laufzeit auf eine andere Iconliste umgeschaltet werden. Dazu erhält jede Iconliste einen Integerwert als ID, der beim Umschalten anzugeben ist. Jedem Icon der Iconliste wird ebenfalls ein Integerwert als ID zugeordnet. Um bestimmte Icons zu aktivieren oder zu deaktivieren werden *Iconsets* verwendet. Ein *Iconset* besteht aus einer Liste mit den IDs aller Icons, die in der aktuellen Iconliste als aktiv dargestellt werden sollen. Alle anderen Icons werden inaktiv abgebildet und können nicht angeklickt werden. Damit ein *Iconset* durch FIX angesprochen werden kann, wird auch ihm eine ID zugeordnet. Durch die Möglichkeit, Icons zu deaktivieren kann die Anzahl der Iconlisten gering gehalten werden, und es werden nur einige universelle Iconlisten benötigt. Das bedeutet, dass die Iconlisten innerhalb einer Anwendung selten oder gar nicht wechseln. Daraus entstehen zwei Vorteile:

- Die IconBox muss nicht jedesmal neu belegt werden, weil sich die Iconliste geändert hat. Dies verhindert ein Flackern auf dem Bildschirm und der Programmablauf wird ruhiger.
- Die Position der Icons ändert sich nicht. Der Benutzer findet gleiche Icons immer an gewohnten Positionen, was die Bedienung erleichtert.¹

Abb. 11 Ermittlung aktiver Icons



Defaultmäßig wird nach erfolgreichem Verbindungsaufbau die Iconliste mit der ID 1 eingestellt. Diese Einstellung kann durch die FIX-Anwendung explizit durch einen Funktionsaufruf geändert werden. Als *Iconset* ist zu Beginn das Set mit der ID 0 eingestellt. Während eine FIX-Anwendung läuft, ändert sich dieses Set laufend. Die Anwendung teilt FIX/Win hierfür zwei Set-ID's mit: eine ID für das *FIX-Iconset* und eine für das *User-Iconset*. Die Vereinigungsmenge der Icons beider Sets wird dann für die endgültige Darstellung verwendet. Das *FIX-Iconset* ist von der Situation abhängig, in der sich die Anwendung gerade befindet. Seine ID kann aus [Tabelle 3 auf Seite 46](#) entnommen werden. Das *User-Iconset* wird durch die ID bestimmt, die innerhalb der Objektbeschreibungsdateien der FIX-Anwendung an einen

1. Dies sollte auch beachtet werden, wenn mehrere ähnliche Iconlisten verwendet werden.

Menüpunkt oder ein Feld gebunden ist. Wenn die Vereinigungsmenge der beiden Iconsets IDs enthält, die nicht in der Iconliste existieren, dann werden diese Werte ignoriert (9 und 13 im obigen Beispiel). Wenn die Iconliste Icons definiert, die nicht in der Vereinigungsmenge vorkommen, dann werden diese nicht aktiviert (7,3,8 und 4 im obigen Beispiel). Alle Icons der Iconliste, deren ID auch in der Vereinigungsmenge vorkommt, werden aktiviert und können angeklickt werden (1,5 und 11 im obigen Beispiel).

Tabelle 3: **FIX**-Iconset ID's

ID	Situation ^a
0	Leeres Iconset
1	Maske perform
2	Submaske perform
3	Rollmaske perform
4	Tabellenmaske perform
5	Selo
6	Choice
7	Menü
11	Maske present
12	Submaske present
13	Rollmaske present
14	Tabellenmaske present
20	Helptext
21	Maske perform, mit Variante
22	Submaske perform, mit Variante
23	Rollmaske perform, mit Variante
24	Tabellenmaske perform, mit Variante
31	Maske present, mit Variante
32	Submaske present, mit Variante
33	Rollmaske present, mit Variante
34	Tabellenmaske present, mit Variante
42	“Weiter“-Zustand
85	Selo Startwerterfassung
101	Return-Message
102	“Ja/Nein“-Message
257	Maske perform, mit Selo
258	Submaske perform, mit Selo
259	Rollmaske perform, mit Selo
260	Tabellenmaske perform, mit Selo
277	Maske perform, mit Selo und Variante
278	Submaske perform, mit Selo und Variante
279	Rollmaske perform, mit Selo und Variante
280	Tabellenmaske perform, mit Selo und Variante

a. siehe FIX-Handbuch

Um den Aufbau der IconBox zu ändern, ist zum einen eine Erweiterung der zur Verfügung stehenden Bitmaps und zum anderen eine Änderung der Icondefinitionsdatei notwendig.

3.1.1 Aufbau der Icondefinitionsdatei

In der Icondefinitionsdatei `config/stdicon.fix` werden alle Iconlisten und Iconsets für ein Gruppenverzeichnis definiert. Die Datei ist wie folgt aufgebaut:

```

# Kommentar
ICONLIST <LISTID>
  <ICONID> <EVENTFOLGE> <BITMAPDATEI> <HINWEISTEXT> ;
  oder SPACE <SIZE> ;
  oder SPACE <SIZE> <BITMAPDATEI>;
  ...
END
...
ICONSET <SETID>
  <ICONID> ;
  ....
END
...

```

Für Kommentare in der Definitionsdatei ist das Zeichen # zu verwenden. Der gesamte Text hinter diesem Zeichen bis zum Zeilenende ist Kommentar.

Die Definition einer Iconliste beginnt mit dem Schlüsselwort **ICONLIST**. Danach ist die ID der Liste anzugeben. Für IDs von Iconlisten sind ganze positive Zahlen von 1 bis 2147483647 zu verwenden. Die ID 0 wird intern für die leere Iconliste verwendet. Für jedes Icon und jeden Zwischenraum folgt dann eine Definition mit Werten, die durch ein Semikolon abzuschließen ist. Die Icons und die Zwischenräume werden später in der gleichen Reihenfolge dargestellt, in der sie in der Iconliste definiert sind.

Die Definition eines Icons besteht aus der Angabe einer Icon-ID, einer Folge von maximal fünf FIX-Events, einer Bitmapdatei zur Darstellung des Icons und einem Hinweistext, der als Tooltip angezeigt wird. Als Icon-ID können alle positiven ganzen Zahlen bis 65535 verwendet werden. Der Name der Bitmapdatei ist ohne Pfad anzugeben. Die Datei wird im Unterverzeichnis `LookAndFeel/IconBox-LAF[0-3]` des Gruppenverzeichnisses erwartet. Wenn sie dort nicht zu finden ist, wird versucht, sie aus dem Unterverzeichnis `LookAndFeel/IconBox-LAF` des FIX/Win-Arbeitsverzeichnisses, in dem sich die FIX/Win-Standardicons befinden, zu laden. Enthält der Hinweistext Leerzeichen, muss er in " " eingefaßt werden. Die Zeichen werden gemäß Windows ANSI Code interpretiert. Als Eventfolge sind ein bis fünf durch Leerzeichen getrennte Eventbezeichner anzugeben. Die Symbole, die FIX-Events bezeichnen, sind aus [Tabelle 4 auf Seite 47](#) zu entnehmen. Zusätzlich zu diesen Symbolen besteht die Möglichkeit einen beliebigen Wert als Event anzugeben. Dazu ist statt eines Symbols der Wert hinter einer Tilde anzugeben. Zum Beispiel definiert `~303` den Event mit dem Code 303.

Ein Zwischenraum zwischen zwei Icons kann durch Verwendung des Schlüsselwortes **SPACE** anstelle eines Icons definiert werden. Hinter diesem Schlüsselwort ist die Größe des Zwischenraums in Pixeln anzugeben. In der Mitte des Leerraums wird dann eine Linie gezeichnet. Alternativ kann zusätzlich der Name einer Bitmap angegeben werden. Die Bitmap wird aus dem Unterverzeichnis `LookAndFeel/IconBox-LAF[0-3]` geladen und auf die angegebene Pixelbreite gestreckt oder gestaucht. Wenn für die IconBox statt der horizontalen die vertikale Darstellung gewählt wird, dann wird versucht eine andere Bitmap für diese Darstellung zu laden. Dazu wird der angegebene Name um den Postfix `V_` erweitert (also beispielsweise `V_space.bmp` statt `space.bmp`). Existiert die Datei nicht, wird die angegebene Datei zur Darstellung des Zwischenraums verwendet. In beiden Fällen wird der angegebene Abstand als Pixelhöhe interpretiert und das Bitmap wird entsprechend gestreckt oder gestaucht.

Zum Abschluss der Definition einer Iconliste ist das Schlüsselwort **END** anzugeben.

Tabelle 4: FIX/Win-Eventbezeichnungen

Tastenevent	Bedeutung	Tastenevent	Bedeutung
HP	Help	PR	Previous Field
PL	Page Left	EN	End
kl	Key Left	kr	Key Right
kh	Key Home	DL	Delete
TB	Tab	ku	Key Up
kd	Key Down	RT	Return
PD	Page Down	PU	Page Up
CI	Character Insert	CD	Character Delete
WI	Word Insert	WD	Word Delete
LI	Line Insert	LD	Line Delete

Tabelle 4: FIX/Win-Eventbezeichnungen

Tastenevent	Bedeutung	Tastenevent	Bedeutung
CE	Clear Entry	BT	Back Tab
ST	Start	PT	Print
MD	Mode	f1	Funktionstaste f1
f2	Copy Field	f3	Jump Field
f4	Insert	f5	Detail
f6	Add	f7	Delete
f8	Key First	f9	Key Selection
f0	Key Last	g1	Rebuild
g2	Funktionstaste g2	g3	Funktionstaste g3
g4	Funktionstaste g4	g5	Funktionstaste g5
g6	Funktionstaste g6	g7	Funktionstaste g7
g8	Funktionstaste g8	g9	Funktionstaste g9
g0	Funktionstaste g0	h1	Funktionstaste h1
h2	Funktionstaste h2	h3	Funktionstaste h3
h4	Funktionstaste h4	h5	Funktionstaste h5
h6	Funktionstaste h6	h7	Funktionstaste h7
h8	Funktionstaste h8	h9	Funktionstaste h9
h0	Funktionstaste h0	-	-

Ein Iconset definiert eine Menge von Icons, die - sofern in der aktiven Iconliste enthalten - als aktiv dargestellt werden sollen. Hierzu werden die IDs der Icons hinter der ID des Iconsets angegeben. Als Trennzeichen ist ein Semikolon zu verwenden. Die Liste ist durch das Schlüsselwort END abzuschließen. Als ID eines Iconsets ist eine ganze positive Zahlen im Bereich von 1 bis 2147483647 zu verwenden. Die ID's 1-1000 sind für die FIX-Iconsets reserviert. Alle anderen Werte können für User-Iconsets verwendet werden. Die ID 0 wird intern für das leere Iconset verwendet.

3.1.2 Vorgehensweisen bei Änderungen

Dieser Abschnitt erläutert einige Vorgehensweisen für Änderungen am Aufbau der IconBox.

Entfernen von Icons

In der Standardkonfiguration werden zu fast allen FIX-Events Icons angeboten. Zur besseren Übersicht für den Anwender können einige seltener verwendete Icons aus der IconBox entfernt werden. Hierzu sind aus der Icondefinitionsdatei alle diesbezüglichen Icondefinitionen innerhalb der `ICONLIST`-Definition zu entfernen. Aus den Iconsets müssen die entsprechenden Icon-ID's nicht entfernt werden, da diese ignoriert werden, wenn sie in der Iconliste nicht enthalten sind.

Hinzufügen von Icons

In FIX-Anwendungen gibt es oft die Möglichkeit, von einem bestimmten Feld einer Maske aus durch eine Taste in ein anderes Modul zu verzweigen. Hier bietet es sich an, ein Icon in FIX/Win anzulegen, das nur dann aktiv ist, wenn das entsprechenden Feld aktiv ist. Dazu ist wie folgt vorzugehen:

- Zuweisung einer Iconset-ID an das Feld: In der Beschreibungsdatei der entsprechenden Maske ist für das Feld das Attribut `iconset <Nummer>` einzutragen.
- Das Icon ist in der zu diesem Zeitpunkt aktuellen Iconliste zu definieren. Dabei ist eine Bitmapdatei anzulegen, die als Abbildung dient.
- Es ist ein neues Iconset anzulegen, das das in der Iconliste neu angelegte Icon aktiviert. Die Nummer des Iconsets muss der Nummer entsprechen, die in der Beschreibungsdatei vergeben wurde. Im Iconset selbst muss die Nummer des Icons angegeben werden, das in die Iconliste eingefügt wurde.

Um Informationen über die aktuellen Werte einer laufenden FIX-Anwendung zu erhalten, kann der Menüpunkt *Iconinfo* aus dem Diagnose-Menü (siehe *Möglichkeiten des Diagnosemenüs* auf Seite 77) aufgerufen werden.

Hinzufügen von alternativen Iconlisten

In manchen FIX-Anwendungen kann es sinnvoll sein, in bestimmten Situationen nicht nur den Zustand von Icons zu ändern, sondern die komplette Iconliste durch eine andere zu ersetzen. So ist beispielsweise eine FIX-Anwendung vorstellbar, die beim Betreten einiger weniger Masken eine umfangreiche Iconliste und beim Betreten von Standardmasken eine kurze und einfache Iconliste verwendet.

Beim Hinzufügen von Iconlisten müssen folgende Schritte befolgt werden:

- In der Icondefinitionsdatei ist eine neue ICONLIST-Definition mit einer neuen ID zu erstellen.
- Verwendet die Datei neue Icons, müssen die vorhandenen Iconsets entsprechend angepasst werden, so dass die Icons aktiviert werden, wenn die FIX-Anwendung sich in der jeweiligen Situation befindet. Die Iconsets müssen dabei die entsprechenden Icon-ID's der Iconlisten enthalten, da sie mit beiden Listen zusammenarbeiten. Hier sei noch einmal darauf hingewiesen, dass im Iconset vorhandene Icon-ID's, die nicht in der aktuellen Iconliste enthalten sind, ignoriert werden.
- Die FIX-Anwendung ist so zu gestalten, dass sie beim Betreten der Masken jeweils die richtige Iconlist einschaltet (siehe FIX Handbuch, Funktion `fx_iconlist()`).

Beispiel zur Icondefinition

In einer Anwendung soll ein neues Icon hinzugefügt werden, mit dem in den "Kundenstamm" verzweigt werden kann. Innerhalb der FIX-Anwendung wird dies durch die Taste f5 erreicht, wenn das Feld "Name des Kunden" aktiv ist. Da die FIX-Anwendung die Iconliste nicht umschaltet, ist zu diesem Zeitpunkt die Iconliste mit der ID 1 aktiv.

Als erstes sind eine freie Iconset-ID und eine freie Icon-ID auszuwählen, die von FIX noch nicht verwendet werden:

```
Iconset ID: 10000
Icon ID: 1000
```

Nun muss in der Anwendung die Iconset-ID dem entsprechenden Feld, bei dessen Betreten das Icon aktiv werden soll, zugeordnet werden; in diesem Fall "Name des Kunden". Hierzu ist in der Beschreibungsdatei der entsprechenden Maske der Eintrag

```
iconset 10000
```

in der Felddefinition zu ergänzen. Als nächstes ist die Icondefinitionsdatei zu bearbeiten. Da die Standard-Iconliste mit der ID 1 beim Betreten des Feldes aktiv ist, ist diese Iconliste um ein Icon zu erweitern. Hierzu ist die folgende Zeile innerhalb der ICONLIST 1 -Definition zu plazieren:

```
1000 f5 kunde.bmp "Verzweigen zum Kundenstamm"
```

Damit das Icon 1000 aktiv wird, ist ein neues Iconset anzulegen. Hierzu ist der folgende Eintrag zu machen:

```
ICONSET 10000
1000;
END
```

Als letzter Schritt ist nun noch die Bitmap `kunde.bmp` anzulegen. Betritt nun der Benutzer das Feld, in dem er in den "Kundenstamm" verzweigen kann, wird FIX/Win mitgeteilt, dass das User-Iconset 10000 aktiv wird. Nun werden alle in diesem Iconset vorhandenen Icons - in diesem Fall nur das Icon 1000 - als aktiv eingeblendet und können angeklickt werden. Da Iconset 1000 ein User-Iconset ist, wird das Icon zusätzlich zu denen des aktuellen FIX-Iconsets eingeblendet. Beim Anklicken des Icons wird dann das Event `f5` an die FIX-Anwendung gesendet, worauf diese in den "Kundenstamm" verzweigt.

3.2 Aussehen der IconBox

Das Aussehen wird durch die Dateien in einem der Unterverzeichnisse

LookAndFeel\IconBox-LAF,
 LookAndFeel\IconBox-LAF-0,
 LookAndFeel\IconBox-LAF-1,
 LookAndFeel\IconBox-LAF-2,
 LookAndFeel\IconBox-LAF-3

bestimmt. Aus welchem Unterverzeichnis die entsprechende Datei gelesen wird, wird durch den eingestellten Bildstil bestimmt. Zum Auffinden der Datei wird das in [Verfahren zum Einlesen von Dateien](#) auf Seite 44 beschriebene Verfahren verwendet.

3.2.1 Einstellungen über Ressourcen

Zur Darstellung der IconBox wird die Datei

`iconbox.frc`

gelesen.

Dort können Einstellungen im FIX/Win-Ressourcenformat (siehe [Allgemeines zu Ressourcen](#) auf Seite 79) vorgenommen werden. Die folgende Tabelle zeigt die möglichen Einstellungen.

Tabelle 5: Ressourcen der IconBox

Ressource	Beschreibung	Defaultwert
IconSizeX	Breite eines Icons in Pixeln	22
IconSizeY	Höhe eines Icons in Pixeln	18
FrameX	Breite des Rahmens links und rechts in Pixeln	3
FrameY	Höhe des Rahmens oben und unten in Pixeln	3
InsetX	Abstand der Icons vom Fensterrahmen links und rechts	2
InsetY	Abstand der Icons vom Fensterrahmen oben und unten	2
Lines	Anzahl Zeilen in der IconBox	1
LineSpacing	Abstand zwischen den Zeilen in Pixeln	1
Like97	Icons mit Mouse-Over Effekt verwenden	true
Background	Hintergrundfarbe der IconBox	sys(menu)
SepLine1	Linienfarbe 1 der Trennlinie	RGB_808080
SepLine1	Linienfarbe 2 der Trennlinie	RGB_FFFFFFFF
TransparentColor	Farbe, die als durchsichtig angesehen wird	RGB_000000
TooltipBackground	Hintergrundfarbe eines Tooltips der IconBox	RGB_FFFFBE
TooltipTimeout	Timeout in Millisekunden, nachdem ein Tooltip angezeigt wird.	3500

3.2.2 Bitmaps für Icons

Um ein neues Icon zu erstellen, muss eine neue Bitmap angelegt werden. Beim Erzeugen neuer Bitmaps sollten folgende Dinge beachtet werden:

- Als Format muss Windows-bmp verwendet werden.
- Damit das Bitmap nicht skaliert wird, sollte die in der Ressourcendatei definierte Größe verwendet werden (`IconSizeX`, `IconSizeY`). Wird dort keine Größe definiert, dann gilt der Standardwert von 22x18.
- Der Hintergrund der Bitmap sollte zum Hintergrund der IconBox passen (Eintrag `Background` in `iconbox.frc`).
- Die als transparent definierte Farbe (Eintrag `TransparentColor` in `iconbox.frc`) sollte beachtet werden.
- Das Aussehen der Bitmap sollte zu den Bitmaps für die Rahmen passen (`frm...bmp`).
- Der Name der Bitmap-Datei ist in der Icondefinitionsdatei zu dem Icon zu hinterlegen (ohne Pfadangabe).
- Das Verzeichnis ergibt sich aus dem Gruppenverzeichnis und dem Bildstil. Es wird für jedes Icon getrennt bestimmt. Das bedeutet, dass nicht alle Icons eines Stil im gleichen Verzeichnis zu finden sind. Einige können sich

auch im stilunabhängigen Verzeichnis `LookAndFeel\IconBox-LAF` oder in einem Unterverzeichnis von `FIX/Win` selbst befinden. Weitere Informationen zum Laden der Icons sind im Abschnitt [Aufbau eines Gruppenverzeichnisses](#) auf Seite 41 zu finden.

- Für die Darstellung im nicht aktiven (nicht anklickbaren) Zustand kann eine Bitmap hinterlegt werden. Als Name ist der Basisname mit dem Postfix `gray_` zu verwenden (z.B. `gray_KUNDE.bmp` enthält die nicht aktive Darstellung für `KUNDE.bmp`). Existiert diese Bitmap nicht, dann lädt `FIX/Win` die originale Bitmap ein zweites Mal und setzt alle Farben auf grau.

3.2.3 Bitmaps für Rahmen

Die Rahmen der Icons werden ebenfalls durch Bitmaps dargestellt. Für sie sind feste Namen zu verwenden. Dabei gibt es zwei Gruppen: Einmal die Rahmen mit Mouse-Over-Effekt (Ressource `Like97==true`) und einmal ohne (Ressource `Like97==false`).

Tabelle 6: Bitmaps für Iconrahmen

Datei	Rahmen
<code>frmActive.bmp</code>	Rahmen für Icon gedrückt
<code>frmInactive.bmp</code>	Rahmen für Icon normal
<code>frmDisabled.bmp</code>	Rahmen für Icon nicht aktiv (nicht anklickbar)
<code>frmActive97</code>	Rahmen für Icon normal mit <code>Like97==true</code>
<code>frmInactive97</code>	Rahmen für Icon nicht aktiv (nicht anklickbar) mit <code>Like97==true</code>
<code>frmSelect97</code>	Rahmen für Icon mit Maus über dem Icon mit <code>Like97==true</code>

Für die vertikale Darstellung der IconBox besteht die Möglichkeit einen weiteren Satz Bitmaps zu definieren. Dazu ist dem Dateinamen der Postfix `v_` voranzustellen. Existiert diese Bitmap nicht, dann wird die gleiche Bitmap wie bei der horizontalen Darstellung verwendet. Ansonsten wird zum Einlesen der Bitmaps für die Rahmen die gleiche Suchreihenfolge durchlaufen, wie beim Laden der anderen Bitmaps (siehe [Aufbau eines Gruppenverzeichnisses](#) auf Seite 41).

Die Größe einer Bitmap für einen Rahmen ergibt sich aus den Ressourcen nach folgender Formel:

$$\text{IconSizeX} + \text{FrameX} * 2 \times \text{IconSizeY} + \text{FrameY} * 2$$

Das Icon wird innerhalb des Rahmens auf der relativen Position `FrameX`, `FrameY` gezeichnet. Wenn die Bitmaps für die Rahmen andere Größen besitzen, dann werden sie auf die entsprechende Größe skaliert.

3.3 Umschalten des Mauscursor bei maussensitiven Masken

Da in der Umstellungsphase von Anwendungen nur bestimmte Masken mausbedienbar sind, soll dies dem Anwender sichtbar gemacht werden. Dies geschieht durch einen andersartigen Mauscursor, der eingeblendet wird, sobald eine Maske aktiv wird, die mausbedienbar ist.



Im ersten Ansatz liegt es nahe, eine Maske dann als mausbedienbar darzustellen, wenn die Beschreibungsdatei den Eintrag `enable mouse` enthält. Dies bedingt jedoch eine Protokolländerung, weshalb von dieser Möglichkeit abgesehen wird. Aus diesem Grund wird das Iconset zur Bestimmung der Mausbedienbarkeit herangezogen. Ein Objekt wird in folgenden Fällen als mausbedienbar dargestellt:

1. Das `FIX`-Iconset hat den Wert 5. Dies ist bei einem Selo der Fall.
2. Das `FIX`-Iconset hat den Wert 6. Dies ist bei einer Choice der Fall.
3. Das `FIX`-Iconset hat den Wert 7. Dies ist bei einem Menue der Fall.
4. Eines beiden oberen Bits (6 und 7) des `USER`-Iconsets ist gesetzt.
Diese Bits werden bei Bestimmung des Iconsets selbst ausgeblendet.

Will man also eine Maske (oder ein Menue) als mausbedienbar darstellen, dann muss man zu den Angaben der User-Iconsets der Felder den Wert 16777216 hinzu addieren, um Bit 6 zu setzen. Dabei kann durchaus ausgenutzt werden, dass die Zuordnung des Iconsets feldweise vorgenommen werden muss. Soll z.B. eine Maske erst nach Verlassen des ersten Feldes als mausbedienbar dargestellt werden, dann kann man das Iconset des ersten Feldes im Originalzustand belassen. (Vorsicht: Die Darstellung der Mausbedienbarkeit durch einen anderen Mauscursor sollte nicht mit der tatsächlichen Mausbedienbarkeit gleichgesetzt werden !)

Um den Mauscursor zur Darstellung der Mausbedienbarkeit (eine Hand, mit der Darstellung eines Feldwechsels) gezielt darzustellen, wird dies über die Ressource `ShowJumpCursor` geregelt. Sie besteht aus einer Integer Zahl, welche einen Bitvector darstellt. Dabei haben die Bits folgende Bedeutung:

BIT	WERT	Bedeutung
0	1	Darstellung des Mauscursors abhängig vom USER-Iconset
1	2	Darstellung des Mauscursors in Selos
2	4	Darstellung des Mauscursors in Choices
4	8	Darstellung des Mauscursors in Menues

Setzt man diese Ressource beispielsweise auf 5, dann wird der Mauscursor in Choices und, abhängig vom USER-Iconset, in Masken und Menues eingeschaltet.

4 Erstellen einer eigenen Tastenbelegung

Im Normalfall wird die in der Datei `config/stdkey.fix` hinterlegte Standardtastenbelegung verwendet. Soll jedoch mit einer einheitlichen oder anderen Tastenbelegung für FIX/Win und Terminal gearbeitet werden, so besteht die Möglichkeit, FIX/Win an die Terminaltastatur anzupassen¹.

Um die Tastenbelegung von FIX/Win zu ändern, ist eine eigene Tastenbelegungsdatei zu erstellen. Dazu kann die Datei `config/stdkey.fix` aus dem Arbeitsverzeichnis von FIX/Win in das Unterverzeichnis `config` des Gruppenverzeichnisses kopiert und angepasst werden. Jede Zeile der Datei definiert die Belegung einer Taste. Eine Zeile hat folgendem Aufbau:

<Tastename> <Normal> <Shift> <Strg> <Shift-Strg>

<Tastename> bezeichnet die Taste, die auf der Tastatur oder an der Maus gedrückt wird. Die anderen Werte geben FIX-Events an, die gesendet werden, wenn die Taste allein, zusammen mit Shift, zusammen mit Strg und zusammen mit Shift und Strg gedrückt wird. Die Tastensymbole für diese Events sind dabei aus der Tabelle [FIX/Win-Eventbezeichnungen](#) auf Seite 47 zu entnehmen. Wie bei der Icondefinitionsdatei kann auch hier ein Eventcode statt des Eventsymbols angegeben werden, indem eine Tilde vorangestellt wird. Um zu kennzeichnen, dass eine Taste unbelegt ist, ist '-' anzugeben. Die Tastennamen sind in [Tabelle 7](#) aufgelistet. Wird eine Taste aus dem Bereich "A-Z" oder "0-9" belegt, dann wird die normale Belegung und die Belegung mit Shift ignoriert, da diese Tasten fest den normalen Zeichen zugeordnet sind. Ebenso verhält es sich mit der linken Maustaste. Diese kann nur in einer Belegung mit der Strg-Taste und/oder der Shift-Taste verwendet werden, da ihre Funktion ohne diese Modifiertasten bereits von FIX/Win genutzt wird. Wird eine Maustaste zweimal kurz hintereinander gedrückt (Doppelklick), dann wird der zweite Klick so interpretiert, als ob die Shift-Taste zusätzlich gedrückt wird. Wird einer Maustaste oder dem Doppelklick auf eine Maustaste kein Event zugeordnet, dann wird statt dessen ein BT_...-Event gesendet (siehe hierzu auch [Menüpunkt "Kopiermodus ..."](#) auf Seite 37). Die folgende Tabelle zeigt eine Zuordnung von Maustaste zu Event:

Maustaste	Klick	Doppelklick
LBUTT	BT_LEFT	BT_LEFTDBL
MBUTT	BT_MIDDLE	BT_MIDDLEDBL
RBUTT	BT_RIGHT	BT_RIGHTDBL

Eine weitere Besonderheit tritt bei der Belegung der Tasten Strg-V und Strg-C auf. Werden diese Tasten nicht mit einem Event belegt, dann wird beim Auslösen von Strg-C der aktuelle Feldinhalt in die Zwischenablage kopiert und beim Auslösen von Strg-V die Zwischenablage als Events an die FIX-Anwendung gesendet. Diese Funktionalität ist jedoch mit Vorsicht anzuwenden, da sie nicht mit dem Einfügen von Text in ein Feld gleichzusetzen ist. Was durch das Senden einer Tastensequenz tatsächlich passiert ist vom aktuellen Zustand der FIX-Anwendung abhängig. Wird sie beispielsweise versehentlich abgesetzt während der Benutzer sich in einem Menü oder einem Selo befindet können dadurch Menüpunkte ausgelöst werden, deren Aufruf der Benutzer nicht beabsichtigt hat und die unter Umständen je nach Anwendung großen Schaden anrichten. Werden die Tasten in der Tastenbelegungsdatei mit einem Event belegt, dann entfällt die Copy&Paste Funktionalität und es wird statt dessen das definierte Event gesendet..

Tabelle 7: FIX/Win-Tastenbezeichner

Tastename	Taste auf deutscher MF-II Tastatur
END	Ende
HOME	Pos1
INSERT	Einfg
DELETE	Entf
LEFT	Pfeil links
RIGHT	Pfeil rechts
UP	Pfeil hoch
DOWN	Pfeil runter
PRIOR	Bild hoch
NEXT	Bild runter
BACK	rückwärts Entfernen
TAB	Tabulator Taste

1. soweit eine PC-Tastatur dies erlaubt

Tabelle 7: FIX/Win-Tastenbezeichner

Tastename	Taste auf deutscher MF-II Tastatur
F1 ... F12	Funktionstasten
NUMPAD0 ... NUMPAD9	Nummernblock 0-9
MULTIPLY	* Nummernblock
ADD	+ Nummernblock
SUBTRACT	- Nummernblock
DIVIDE	/ Nummernblock
DECIMAL	, Nummernblock
RETURN	Eingabetaste
LBUTT	linke Maustaste
MBUTT	mittlere Maustaste
RBUTT	rechte Maustaste
APPS	Menütaste (rechts neben Windows-Taste)
WHEELFW	Mausrad vorwärts
WHEELBW	Mausrad rückwärts
ESCAPE	ESC-Taste
A..Z	Buchstabentasten
0..9	Zahlentasten

Bei der Angabe von neuen Tastenbelegungen in der Definitionsdatei sollte darauf geachtet werden, dass eine entsprechende Bitmap als Tasten-Label vorhanden ist (siehe nächster Abschnitt). Wird ein FIX-Event mehreren Windows-Tasten zugeordnet, so wird immer die letzte Definition zur Ermittlung des Tasten-Labels herangezogen.

4.1 Tasten-Label

In der Statuszeile von FIX-Programmen werden oft Tastenhinweise ausgegeben. Unter FIX/Win sind diese *Tasten-Labels*¹ anklickbar und senden das entsprechende Event zur FIX-Anwendung. Das Aussehen der Tasten-Labels wird durch Bitmaps bestimmt. Diese Bitmaps werden aus den gepackten Bitmaps `keyBitmap-M.bmp`, `keyBitmap-L.bmp` und `keyBitmap-H.bmp` für die drei verschiedenen Bildschirmgrößen von FIX/Win gelesen (-M - Medium, -L - Large, -H - Huge). Die Dateien werden je nach Bildstil aus einem der Unterverzeichnisse

```
LookAndFeel/FixwinLAF
LookAndFeel/FixwinLAF-1
LookAndFeel/FixwinLAF-2
LookAndFeel/FixwinLAF-3
LookAndFeel/FixwinLAF-4
```

des Gruppenverzeichnisses gelesen. Die Bitmaps entstehen durch das Packen der Bitmaps aus den Unterverzeichnissen von `LookAndFeel-src`. Hier existiert für jeden Stil ebenfalls ein Unterverzeichnis mit dem gleichen Namen. Zur Ablage der Bitmaps enthalten diese Unterverzeichnisse wiederum Unterverzeichnisse, die aus dem Namen der gepackten Bitmaps gebildet werden.

Beispiel:

```
LookAndFeel-src/FixwinLAF-2/keyBitmap-M
```

enthält die ungepackten Bitmaps für Stil 2 in der Größe M. Sie werden nach

```
LookAndFeel/FixwinLAF-2/keyBitmap-M.bmp
```

gepackt (Weitere Informationen dazu sind in [Aufbau eines Gruppenverzeichnisses](#) auf Seite 41 zu finden).

Der Dateiname einer Bitmap für ein Tasten-Label hat in diesen Verzeichnissen die Form:

```
<Name>_<Modifer>.bmp
```

1. Im FIX Handbuch als Tastenbeschriftung bezeichnet.

Dabei ist <Name> der Namen der entsprechende Taste (Tabelle 7, Seite 53) und <Modifier> einer der folgenden Werte:

Tabelle 8: Modifikerkürzel

Modifier	Taste
N	Taste normal
S	Taste mit Shift
C	Taste mit Strg
X	Taste mit Shift und Strg

Beispiel:

F10_S.bmp

ist ein Bitmap, das die Taste Shift-F10 darstellt.





Die Bitmaps für die Standardtastenbelegung befinden sich in den entsprechenden Unterverzeichnissen des Arbeitsverzeichnisses von FIX/Win. Wird die Tastenbelegung geändert, müssen eventuell neue Bitmaps angelegt werden. Diese sind in den Unterverzeichnissen von LookAndFeel-src unterhalb des Gruppenverzeichnisses abzulegen. Weiterhin kann das Aussehen bestehender Tasten-Label-Bitmaps durch Anlegen eines Bitmap mit gleichem Namen geändert werden. Erst wenn ein Bitmap nicht im Gruppenverzeichnis gefunden wird, wird das Standardbitmap aus dem Arbeitsverzeichnis von FIX/Win geladen. Da FIX/Win Bitmaps jedoch nur aus den gepackten Dateien im Verzeichnis bmpack ließt, sind nach einer Änderung die entsprechenden Bitmaps mit dem Programm bmpack neu zu packen. Beim Erstellen neuer Bitmaps sind außer der Namensvergabe noch folgende Punkte zu beachten:

- Folgende Größen (in Pixeln) müssen verwendet werden, wenn mit den mitgelieferten Schriften gearbeitet wird:

Tabelle 9:

Breite x Höhe	Größe	Verzeichnis
28x12	Medium	keyBitmap-M
36x16	Large	keyBitmap-L
44x24	Huge	keyBitmap-H

- Bei der Verwendung von Schriftarten aus dem Windowssystem ergibt sich die Höhe aus der Höhe der Zeile - 4 und die Breite aus der Breite der Zeile * 4 - 4. Die Größe der Zelle kann über den Menüpunkt *Diagnose/Zeicheninfo* ermittelt werden.
- Die Farbe der Bitmaps sollte zum Rahmen passen.
- Die Bitmaps sollten die entsprechende PC-Taste darstellen. Hier einige Beispiele zur Darstellung:

	Funktionstaste mit Strg
	Funktionstaste mit Shift
	Funktionstaste mit Shift + Strg
	Cursortaste

- Die Bitmaps müssen im Windows bmp-Format abgelegt werden.

4.2 Rahmen für Tasten-Labels

Die Rahmen für Tasten-Labels werden ebenfalls durch Bitmaps dargestellt, die je nach Bildstil aus einem Unterverzeichnis von `LookAndFeel` gelesen werden. Die folgende Tabelle zeigt die Dateiname und die Größen in Pixeln, die für die Bitmaps zu verwenden sind, wenn die mitgelieferten Schriftarten verwendet werden.

Tabelle 10: Bitmaps für Rahmen

Dateiname	Rahmen	Größe
keyFrmActive-M.bmp	Rahmen für Tasten-Label in angeklicktem Zustand in der Größe Medium	32 x 16
keyFrmInActive-M.bmp	Rahmen für Tasten-Label in normalem Zustand in der Größe Medium	32 x 16
keyFrmActive-L.bmp	Rahmen für Tasten-Label in angeklicktem Zustand in der Größe Large	40 x 20
keyFrmInActive-L.bmp	Rahmen für Tasten-Label in normalem Zustand in der Größe Large	40 x 20
keyFrmActive-H.bmp	Rahmen für Tasten-Label in angeklicktem Zustand in der Größe Huge	48 x 28
keyFrmInActive-H.bmp	Rahmen für Tasten-Label in normalem Zustand in der Größe Huge	48 x 28

Bei der Verwendung von Schriftarten aus dem Windowssystem ergibt sich die Höhe aus der Höhe der Zelle und die Breite aus der Breite der Zeile * 4. Die Größe der Zelle kann über den Menüpunkt *Diagnose/Zeicheninfo* ermittelt werden.

Zur Bestimmung des Verzeichnisses wird das gleiche Verfahren wie zur Bestimmung des Verzeichnisses der gepackten Bitmap verwendet (Weitere Informationen im Abschnitt *Aufbau eines Gruppenverzeichnisses* auf Seite 41).

4.3 Beispiel einer neuen Tastenbelegung

Mit der Taste Strg-X soll eine FIX-Anwendung alternativ verlassen werden können. Auf der Seite von FIX wird ein Verlassen durch das Event *EN* bewirkt. Die Taste muss mit diesem Event belegt werden. Hierzu wird die ins Gruppenverzeichnis kopierte Datei `stdkey.fix` um folgende Zeile erweitert:

```
X -- -- EN --
```

Da das Event *EN* bereits von der Taste End gesendet wird, kommt es zu Konflikten bei der Darstellung von Tasten-Labels. Da FIX/Win immer die letzte Definition zur Darstellung von Tasten-Labels verwendet, kommt es darauf an, ob die neue Definition vor oder hinter der End-Taste steht.

Steht sie dahinter, dann wird sie zur Darstellung für das Tasten-Label herangezogen; in diesem Fall sind neue Bitmaps mit dem Namen `X_C.bmp` (Taste X zusammen mit Strg) in den Unterverzeichnissen

```
LookAndFeel-src/FixwinLAF/keyBitmap-M
LookAndFeel-src/FixwinLAF/keyBitmap-L
LookAndFeel-src/FixwinLAF/keyBitmap-H
```

des Gruppenverzeichnisses anzulegen¹. Die Bitmaps sollten die Taste Strg-X darstellen. Danach müssen die Bitmaps neu gepackt werden.

Steht die Definition von X vor der Taste End, dann wird die End-Taste zur Darstellung des Tasten-Labels herangezogen. Für diese Taste werden bereits von FIX/Win eine Bitmaps bereitgestellt.

1. Das Beispiel geht davon aus, dass die Tasten-Labels in allen Bildstilen gleich dargestellt werden. Andernfalls sind die entsprechenden stilabhängigen Unterverzeichnisse zu verwenden.

5 Verwendung von Semigrafikzeichen

Zur Darstellung von Rahmen und Feldbegrenzern verwendet FIX festgelegte Codes für Semigrafikzeichen. Diese Zeichencodes werden von FIX/Win durch Bitmaps dargestellt. Da FIX/Win drei Bildschirmgrößen und drei Stile unterstützt, ist je Größen-/Stil-Kombination eine gepackte Bitmap vorhanden, die alle Bitmaps für Semigrafikzeichen enthält. Soll das Aussehen der vorhandenen Stile verändert oder um eigene Grafikzeichen erweitert werden, so sind neue Bitmaps anzulegen. Diese sind analog zu den Bitmaps für die Tasten-Labels in den entsprechenden Unterverzeichnissen des Verzeichnisses `LookAndFeel-src` im Gruppenverzeichnis abzulegen und danach mit `bmpack` zu packen. Dabei sind abhängig von Stil und Größe folgende Unterverzeichnisse zu verwenden:

Tabelle 11: Bitmapverzeichnisse für Semigrafikzeichen

Bildstil	Medium	Large	Huge
0	Fixwin-LAF-0/bitmap-M	Fixwin-LAF-0/bitmap-L	Fixwin-LAF-0/bitmap-H
1	Fixwin-LAF-1/bitmap-M	Fixwin-LAF-1/bitmap-L	Fixwin-LAF-1/bitmap-H
2	Fixwin-LAF-2/bitmap-M	Fixwin-LAF-2/bitmap-L	Fixwin-LAF-2/bitmap-H
3	Fixwin-LAF-3/bitmap-M	Fixwin-LAF-3/bitmap-L	Fixwin-LAF-3/bitmap-H

Der Name einer Bitmap ist nach folgendem Format aufgebaut:

xxx-Y.bmp

'xxx' ist die dreistellige Dezimalzahl des Semigrafikcodes. 'Y' bestimmt das Attribut des Zeichens in folgender Weise:

- 0 - normal (NORMAL)
- 1 - invertiert (INVERS)
- 2 - halbhell (LOW)
- 3 - unterstrichen (UNDERLINE)
- 4 - blinkend (BLINK)
- 5 - fett (BOLD)
- 6 - gegraut (GRAYED)
- 7 - unsichtbar (NOTVISIBLE)

Sollte für ein Grafikzeichen keine Bitmap vorhanden sein, dann wird eine Meldung ausgegeben und an der entsprechenden Stelle im FIX-Bildschirm wird ein Fragezeichen eingeblendet. Die folgende Tabelle zeigt die von FIX verwendeten Codes für Semigrafikzeichen. Alle anderen Codes im Bereich von 32-255 können für eigene Zeichen verwendet werden

Tabelle 12: von FIX verwendete Semigrafikzeichen

Zeichen	Attribute	Bitmapdatei	Semigrafikzeichen
0x001	normal	001-0	Hintergrund LED
	normal	000-0	Bitmap für unbekanntes Zeichen
<Leer>	normal	032-0	Leerraum im Grafikmode
(alle	040-[0123]	Menübegrenzer links
)	alle	041-[0123]	Menübegrenzer rechts
-	normal	045-0	Linie horizontal innen
0	normal	048-0	Linie vertikal innen
1	normal	049-0	Ecke unten links innen
2	normal	050-0	T unten innen
3	normal	051-0	Ecke unten rechts innen
4	normal	052-0	T links innen
5	normal	053-0	Kreuz innen
6	normal	054-0	T rechts innen
7	normal	055-0	Ecke oben links innen
8	normal	056-0	T oben innen
9	normal	057-0	Ecke oben rechts innen
<	alle	060-[0123]	Feldbegrenzer Scrollfeld links
>	alle	062-[0123]	Feldbegrenzer Scrollfeld rechts

Tabelle 12: von FIX verwendete Semigrafikzeichen

Zeichen	Attribute	Bitmapdatei	Semigrafikzeichen
O	normal	079-0	Ecke unten links außen
P	normal	080-0	T unten außen
Q	normal	081-0	Ecke unten rechts außen
R	normal	082-0	T links außen
T	normal	084-0	T rechts außen
U	normal	085-0	Ecke oben links außen
V	normal	086-0	T oben außen
W	normal	087-0	Ecke oben rechts außen
X	normal	088-0	Linie oben außen
Y	normal	089-0	Linie links außen
Z	alle	090-[0123]	Feldfüllzeichen Specialfeld
[alle	091-[0123]	Feldbegrenzer links
]	alle	093-[0123]	Feldbegrenzer rechts
^	alle	094-[0123]	Füllzeichen für Displaylänge
_	alle	095-[0123]	Feldfüllzeichen
x	normal	120-0	Linie unten außen
y	normal	121-0	Linie rechts außen
β	alle	223-[0123]	Füllzeichen für Displaylänge Specialfeld

Weiterhin müssen folgende Regeln beim Erstellen von Bitmaps beachtet werden:

- Folgende Größen (in Pixeln) müssen verwendet werden, wenn die mitgelieferten Schriftarten verwendet werden:

Tabelle 13:

Breite x Höhe	Größe
08x16	Medium
10x20	Large
12x28	Huge

- Bei der Verwendung von Schriftarten aus dem Windowssystem ergibt sich die Größe aus der Größe der Zelle. Die Größe der Zelle kann über den Menüpunkt *Diagnose/Zeicheninfo* ermittelt werden.
- Die Hintergrundfarbe der Bitmaps sollte zum entsprechenden Bildschirmstil passen.
- Die Bitmaps müssen im Windows bmp-Format abgelegt sein, Dateiendung .bmp .

5.1 Beispiele zur Erstellung eigener Semigrafikzeichen

Beispiel 1

Eine Maske soll mit einem Firmenlogo versehen werden. Da ein solches Logo i.d.R. aus mehreren Zeichen besteht, wird es zusammengesetzt. Im Layouteditor von FIX werden in der Maske dazu folgende Zeichen im Grafikmodus gezeichnet:

```
abc
def
```

Nun wird mit einem Bitmapeditor das Firmenlogo erstellt. Es sollte in vier Größen erstellt werden, eines für jede Bildschirmgröße von FIX/Win. Die Pixelgröße berechnet sich dabei nach folgendem Verfahren:

breite = Anzahl der Zeichen horizontal * Breite des Fonts der jeweiligen Größe

höhe = Anzahl der Zeichen vertikal * Höhe des Fonts der jeweiligen Größe

Also ergibt sich beispielsweise für die mittlere Bildschirmgröße eine Auflösung von $3 \cdot 8 \cdot 2^{16} = 24 \cdot 32$.

Nachdem das Logo in der jeweiligen Größe erstellt worden ist, ist es auf die entsprechenden Bitmaps mittels *Ausschneiden/Einfügen*-Mechanismus zu übertragen. Das Zeichen "a" hat den Code 97. Also ist ein Bitmap im Gruppenverzeichnis mit dem Namen 097-0.bmp anzulegen, in welches die linke obere Ecke des Logos kopiert wird. Analog ist mit den Bitmaps für die Semigrafik der Codes "b"- "f" zu verfahren.

Beispiel 2

Es soll ein neuer Bildschirmstil entworfen werden. Hierzu sind zum einen die Farbzustimmungstabellen anzupassen und zum anderen die Bitmaps für Semigrafikzeichen. Dazu werden alle Semigrafikzeichen, die dem neuen Stil am ähnlichsten sind, in das entsprechende Unterverzeichnis des Gruppenverzeichnisses kopiert. Diese Bitmaps können nun bearbeitet werden, so dass die FIX/Win-Masken und -Menürahmen den neuen Farbzustimmungen angepasst werden.

6 Packen von Bitmaps

Beim Aufbau der Verbindung und beim Wechsel des Stils oder der Größe werden von FIX/Win die Zeichen zur Darstellung von Semigrafikzeichen und Tasten-Labels aus den gepackten Bitmapdateien aus einem Unterverzeichnis von LookAndFeel geladen. Dort befindet sich für jede Größen-/Stil-Kombination eine Datei mit den eigentlichen Bitmaps (`bitmap-<gr>.bmp`) und eine Datei, welche Strukturinformationen enthält (`bitmap-<gr>.bif`). Weiterhin gibt es je zwei Dateien für jede Stil-Größenkombination der Tasten-Labels (`keyBitmap-<gr>.bmp` und `keyBitmap-<gr>.bif`). Die Dateien werden aus den einzelnen Bitmaps erzeugt, die im Unterverzeichnis LockAndFeel-src liegen. Der Mechanismus, der zum Einlesen verwendet wird, ist im Abschnitt [Aufbau eines Gruppenverzeichnisses](#) auf Seite 41 beschrieben. Das Packen der Bitmaps hat zwei Vorteile

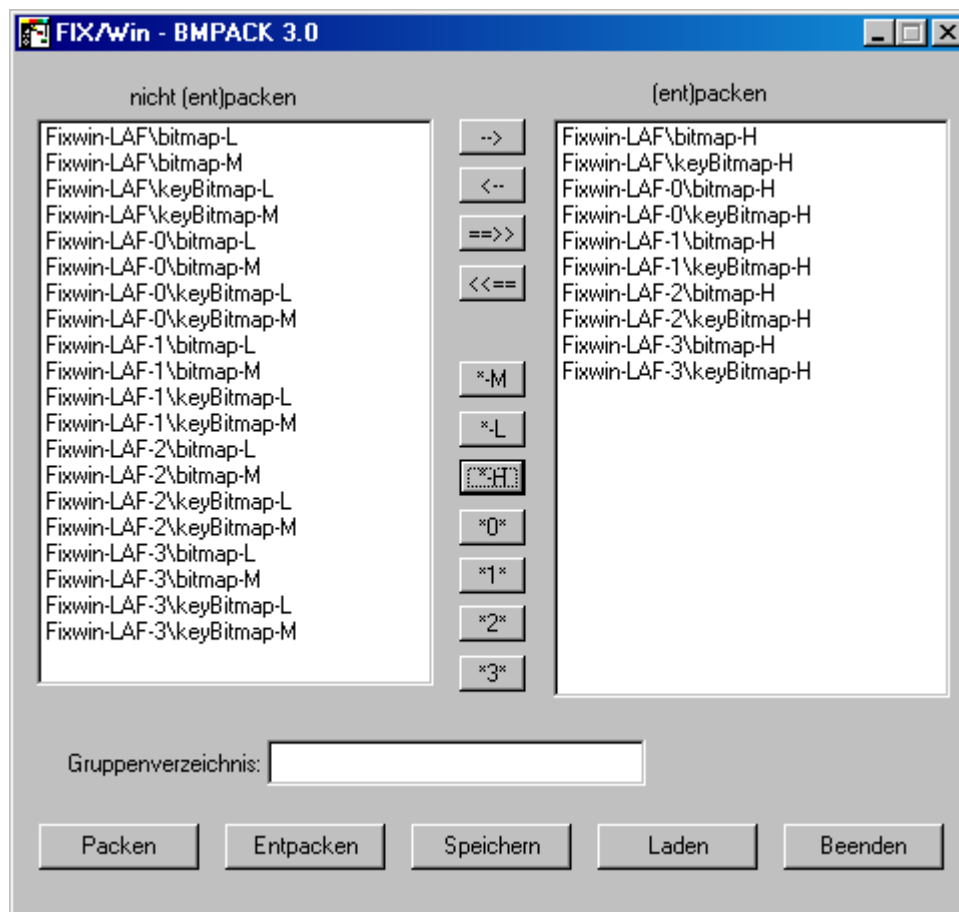
1. In der Runtime Version von FIX/Win werden nur die gepackten Bitmapdateien benötigt, die aufgrund der Blockgröße des Dateisystems erheblich weniger Platz auf der Festplatte benötigen als die vielen kleinen Quellbitmapdateien.
2. Da die Größe der Bitmaps zum Ladezeitpunkt bekannt ist, kann der hierfür reservierte Speicher minimiert werden.

Außerdem ergeben sich folgende Konsequenzen:

- Nach dem Ändern einer Bitmap für ein Semigrafikzeichen oder für ein Tasten-Label muss der entsprechende Bitmap-Satz neu gepackt werden und von FIX/Win nachgeladen werden, um die Änderung sichtbar zu machen.
- Die Runtime-Version von FIX/Win enthält die Bitmaps nur noch in gepackter Form.
- Beim der Installation von FIX/Win beim Anwender sollten die Quellbitmaps nicht mitinstalliert werden. Dazu kann das komplette Verzeichnis LookAndFeel-src sowohl im Gruppenverzeichnis als auch im Arbeitsverzeichnis von FIX/Win weggelassen werden.

Verwendung des Programms bmpack

Mittels des Programms `bmpack` können die Bitmaps von FIX/Win gepackt werden. Folgende Abbildung zeigt die Bedienelemente des Programms.

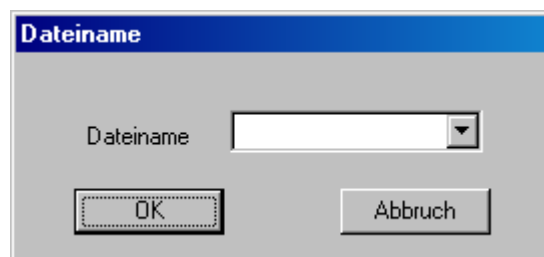


Links befindet sich die Liste der Bitmap-Sätze, die nicht gepackt werden sollen. Rechts befinden sich diejenigen, die gepackt werden sollen. Die Buttons zwischen den Listen dienen zum Verschieben einzelner oder mehrerer Einträge von einer Liste in die andere. Sie haben folgende Funktion:

- > verschiebt die markierten Einträge von links nach rechts
- <- verschiebt die markierten Einträge von rechts nach links
- ==> verschiebt alle Einträge nach rechts
- <== verschiebt alle Einträge nach links
- *_M verschiebt alle Einträge für die Bildschirmgröße Medium nach rechts
- *_L verschiebt alle Einträge für die Bildschirmgröße Large nach rechts
- *_H verschiebt alle Einträge für die Bildschirmgröße Huge nach rechts
- *0* verschiebt alle Einträge für den Bildstil 0 nach rechts
- *1* verschiebt alle Einträge für den Bildstil 1 nach rechts
- *2* verschiebt alle Einträge für den Bildstil 2 nach rechts
- *3* verschiebt alle Einträge für den Bildstil 3 nach rechts

Im Textfeld unter den Listen ist ein Gruppenverzeichnis einzutragen. Danach kann der Button "Packen" betätigt werden. Danach werden die einzelnen Bitmaps für jede Stil-Größenkombination der rechten Seite gepackt. Das Verfahren zum Packen wird im Abschnitt [Verfahren zum Einlesen von Dateien](#) auf Seite 44 beschrieben. Zum Entpacken von der gepackten Bitmaps kann der Button "Entpacken" angeklickt werden. Dabei werden grundsätzlich die entpackten Dateien in einem Unterverzeichnis von `LookAndFeel-src` des Gruppenverzeichnisses abgelegt. Es werden keine Dateien im Arbeitsverzeichnis von `FIX/Win` abgelegt. Stammt eine Datei in der gepackten Datei ursprünglich aus dem Arbeitsverzeichnis, dann wird diese nicht entpackt. Wenn `bmpack` mit dem Kommandozeilenparameter `/UA` (`unpack all`) gestartet wird, dann bewirkt dies, dass auch die Dateien entpackt werden, die aus dem Arbeitsverzeichnis von `FIX/Win` stammen. Sie werden jedoch im Gruppenverzeichnis abgelegt.

Neben dem Button "Packen" befinden sich die Buttons "Laden" und "Speichern". Diese dienen zum Laden und Speichern der Listeninhalte und des Gruppenverzeichniseintrags in einer Konfigurationsdatei.



Dazu ist in der dann eingeblendeten Dialogbox ein Dateiname anzugeben oder ein bestehender durch Drücken des Pfeil-Buttons neben dem Eingabefeld auszuwählen. Das Nachgenerieren bestimmter Bitmap-Sätze kann so schnell vollzogen werden, indem die Einstellungen gespeichert werden und vor dem erneuten Generieren wieder geladen werden.

Die folgenden Kommandozeilenparameter können beim Start von `bmpack` angegeben werden:

- `/G <Gruppenverzeichnis>` Hiermit wird ein Gruppenverzeichnis fest vorgegeben. Es wird dann nicht durch das Laden einer Konfigurationsdatei überschrieben.
- `/C <Configfile>` Beim Programmstart werden die Einstellungen aus der angegebenen Datei gelesen (so wie beim Button "Laden").
- `/A` Nach dem Start des Programms wird das Packen automatisch gestartet.
- `/AU` Nach dem Start des Programms wird das Entpacken automatisch gestartet.
- `/X` Nach dem Packen/Entpacken wird das Programm automatisch beendet.
- `/UA` Dateien die aus dem Arbeitsverzeichnis von `FIX/Win` stammen werden beim Entpacken im Gruppenverzeichnis abgelegt.

Beispiel

Eine Anwendung definiert nur Bitmaps für den Bildstil 2. Beim ersten Aufruf wird `bmpack` ohne Parameter gestartet. Dann werden alle Bitmap-Sätze des Stils 2 in die rechte Liste aufgenommen. Nach dem Eintragen des Gruppenverzeichnisses werden die Einstellungen in der Datei `sty2` gespeichert. Beim Nachgenerieren der gepackten Dateien kann der Vorgang durch folgende Kommandozeilenparameter automatisiert werden:

`tools/bmpack /G mygrpdir /C sty2 /A /X`

Wird `bmpack` über das Entwicklermenü von FIX/Win gestartet, dann können die Parameter in der Ressource `BmPackOptions` hinterlegt werden. Ansonsten ist darauf zu achten, dass `bmpack` im FIX/Win Arbeitsverzeichnis gestartet wird, damit es seine Quellbitmaps findet.

7 Erstellen eigener Farbzuoordnungstabellen

Die Einstellung der Farben wird durch die Datei `coltab.fix` definiert. Sie ist in jedem stilabhängigen Unterverzeichnis `Fixwin-LAF-[0123]` vorhanden. Um die Zuordnung der Farben für einen Stil zu ändern, ist die zum Bildstil gehörende Datei in das gleichnamige Unterverzeichnis des Gruppenverzeichnisses zu kopieren. Die Kopie im Gruppenverzeichnis kann dann den gewünschten Einstellungen angepasst werden. Die Datei besteht aus mehreren Textzeilen, die jeweils eine Farbzuoordnung beschreiben und wie folgt aufgebaut sind:

<Element> <Vordergrundfarbe> <Hintergrundfarbe>

<Element> ist der Name des Elementes und kann aus [Tabelle 15](#) entnommen werden. <Vordergrundfarbe> bestimmt die Vordergrundfarbe und <Hintergrundfarbe> die Hintergrundfarbe eines Elementes. Die möglichen Werte für die Farben können aus [Tabelle 14](#) entnommen werden. Zusätzlich ist es möglich jeden RGB-Wert als Farbe anzugeben. Dazu ist ein Eintrag in der Form

RGB_<RR><GG><BB>

anzugeben. <RR> ist ein zweistelliger hexadezimaler Wert, der den Rotanteil der Farbe im Bereich 00-FF definiert. <GG> definiert den Grünanteil und <BB> den Blauanteil auf die gleiche Weise.

Abgesehen davon können Kommentarzeilen in die Datei aufgenommen werden, indem in die erste Spalte einer Zeile das Zeichen "#" eingetragen wird.

Tabelle 14: : Schlüsselworte für Farbwerte

Farbwerte	
WHITE	BLACK
GRAY1	GRAY2
BLUE1	BLUE2
RED1	RED2
GREEN1	GREEN2
CYAN1	CYAN2
MAGENTA1	MAGENTA2
BROWN	YELLOW

Tabelle 15: Elementnamen in der Farbzuoordnungstabelle

Elementname	Beschreibung
BACKGROUND	Allgemeiner Hintergrund
FIELD_NORMAL	Feld mit Attribut "normal"
FIELD_INVERS	Feld mit Attribut "invers"
FIELD_LOW	Feld mit Attribut "low"
FIELD_UNDERLINE	Feld mit Attribut "underline"
FIELD_BLINK	Feld mit Attribut "blink"
FIELD_GRAYED	Feld mit Attribut "grayed"
FIELD_BOLD	Feld mit Attribut "bold"
FIELD_INVISIBLE	Feld mit Attribut "invisible"
FIELD_LINE1	Feldlinie innen
FIELD_LINE2	Feldlinie außen
FIELD_LINE1_INVERS	Feldlinie innen mit Attribut "invers"
FIELD_LINE2_INVERS	Feldlinie außen mit Attribut "invers"
TABLE_NORMAL	Tabellenfeld mit Attribut "normal"
TABLE_INVERS	Tabellenfeld mit Attribut "invers"
TABLE_LOW	Tabellenfeld mit Attribut "low"
TABLE_UNDERLINE	Tabellenfeld mit Attribut "underline"
TABLE_BLINK	Tabellenfeld mit Attribut "blink"
TABLE_GRAYED	Tabellenfeld mit Attribut "grayed"

Tabelle 15: Elementnamen in der Farbzuordnungstabelle

Elementname	Beschreibung
TABLE_BOLD	Tabellenfeld mit Attribut "bold"
TABLE_INVISIBLE	Tabellenfeld mit Attribut "invisible"
TABLE_LINE1	Tabellenfeld Linie
TABLE_LINE2	Tabellenfeldlinie mit Attribut "invers"
SPECIAL_NORMAL	Special-Feld mit Attribut "normal"
SPECIAL_INVERS	Special-Feld mit Attribut "invers"
SPECIAL_LOW	Special-Feld mit Attribut "low"
SPECIAL_UNDERLINE	Special-Feld mit Attribut "underline"
SPECIAL_BLINK	Special-Feld mit Attribut "blink"
SPECIAL_GRAYED	Special-Feld mit Attribut "grayed"
SPECIAL_BOLD	Special-Feld mit Attribut "bold"
SPECIAL_INVISIBLE	Special-Feld mit Attribut "invisible"
SPECIAL_LINE	Special-Feld Linie
MENU_NORMAL	Menüeintrag mit Attribut "normal"
MENU_INVERS	Menüeintrag mit Attribut "invers"
MENU_LOW	Menüeintrag mit Attribut "low"
MENU_UNDERLINE	Menüeintrag mit Attribut "underline"
MENU_BLINK	Menüeintrag mit Attribut "blink"
MENU_GRAYED	Menüeintrag mit Attribut "grayed"
MENU_BOLD	Menüeintrag mit Attribut "bold"
MENU_INVISIBLE	Menüeintrag mit Attribut "invisible"
MENU_LINE	Menüeintrag Linie
CHOICE_NORMAL	Choice-Eintrag mit Attribut "normal"
CHOICE_INVERS	Choice-Eintrag mit Attribut "invers"
CHOICE_LOW	Choice-Eintrag mit Attribut "low"
CHOICE_UNDERLINE	Choice-Eintrag mit Attribut "underline"
CHOICE_BLINK	Choice-Eintrag mit Attribut "blink"
CHOICE_GRAYED	Choice-Eintrag mit Attribut "grayed"
CHOICE_BOLD	Choice-Eintrag mit Attribut "bold"
CHOICE_INVISIBLE	Choice-Eintrag mit Attribut "invisible"
CHOICE_LINE	Choice-Eintrag Linie
SELO_NORMAL	Selo-Eintrag mit Attribut "normal"
SELO_INVERS	Selo-Eintrag mit Attribut "invers"
SELO_LOW	Selo-Eintrag mit Attribut "low"
SELO_UNDERLINE	Selo-Eintrag mit Attribut "underline"
SELO_BLINK	Selo-Eintrag mit Attribut "blink"
SELO_GRAYED	Selo-Eintrag mit Attribut "grayed"
SELO_BOLD	Selo-Eintrag mit Attribut "bold"
SELO_INVISIBLE	Selo-Eintrag mit Attribut "invisible"
SELO_LINE1	Selo-Eintrag Linie
SELO_LINE2	Selo-Eintrag Linie mit Attribut "invers"
LABEL_NORMAL	Selolabel mit Attribut "normal"
LABEL_INVERS	Selolabel mit Attribut "invers"
LABEL_LOW	Selolabel mit Attribut "low"
LABEL_UNDERLINE	Selolabel mit Attribut "underline"
LABEL_BLINK	Selolabel mit Attribut "blink"
LABEL_GRAYED	Selolabel mit Attribut "grayed"
LABEL_BOLD	Selolabel mit Attribut "bold"

Tabelle 15: Elementnamen in der Farbzuoordnungstabelle

Elementname	Beschreibung
LABEL_INVISIBLE	Selolabel mit Attribut "invisible"
LABEL_LINE	Selolabel Linie
HEAD_NORMAL	Überschrift mit Attribut "normal"
HEAD_INVERS	Überschrift mit Attribut "invers"
HEAD_LOW	Überschrift mit Attribut "low"
HEAD_UNDERLINE	Überschrift mit Attribut "underline"
HEAD_BLINK	Überschrift mit Attribut "blink"
HEAD_GRAYED	Überschrift mit Attribut "grayed"
HEAD_BOLD	Überschrift mit Attribut "bold"
HEAD_INVISIBLE	Überschrift mit Attribut "invisible"
HEAD_LINE	Überschrift Linie
TEXT_NORMAL	Text mit Attribut "normal"
TEXT_INVERS	Text mit Attribut "invers"
TEXT_LOW	Text mit Attribut "low"
TEXT_UNDERLINE	Text mit Attribut "underline"
TEXT_BLINK	Text mit Attribut "blink"
TEXT_GRAYED	Text mit Attribut "grayed"
TEXT_BOLD	Text mit Attribut "bold"
TEXT_INVISIBLE	Text mit Attribut "invisible"
PA_TEXT_NORMAL	Textlabel mit Attribut "normal"
PA_TEXT_INVERS	Textlabel mit Attribut "invers"
PA_TEXT_LOW	Textlabel mit Attribut "low"
PA_TEXT_UNDERLINE	Textlabel mit Attribut "underline"
PA_TEXT_BLINK	Textlabel mit Attribut "blink"
PA_TEXT_GRAYED	Textlabel mit Attribut "grayed"
PA_TEXT_BOLD	Textlabel mit Attribut "bold"
PA_TEXT_INVISIBLE	Textlabel mit Attribut "invisible"
PA_HEADER_NORMAL	Tabellenüberschrift mit Attribut "normal"
PA_HEADER_INVERS	Tabellenüberschrift mit Attribut "invers"
PA_HEADER_LOW	Tabellenüberschrift mit Attribut "low"
PA_HEADER_UNDERLINE	Tabellenüberschrift mit Attribut "underline"
PA_HEADER_BLINK	Tabellenüberschrift mit Attribut "blink"
PA_HEADER_GRAYED	Tabellenüberschrift mit Attribut "grayed"
PA_HEADER_BOLD	Tabellenüberschrift mit Attribut "bold"
PA_HEADER_INVISIBLE	Tabellenüberschrift mit Attribut "invisible"
PA_HEADER_LINE1	Tabellenüberschrift Linie1
PA_HEADER_LINE2	Tabellenüberschrift Linie2
PA_BUTTON_NORMAL	Button mit Attribut "normal"
PA_BUTTON_INVERS	Button mit Attribut "invers"
PA_BUTTON_LOW	Button mit Attribut "low"
PA_BUTTON_UNDERLINE	Button mit Attribut "underline"
PA_BUTTON_BLINK	Button mit Attribut "blink"
PA_BUTTON_GRAYED	Button mit Attribut "grayed"
PA_BUTTON_BOLD	Button mit Attribut "bold"
PA_BUTTON_INVISIBLE	Button mit Attribut "invisible"
PA_BUTTON_LINE1	Button Linie 1
PA_BUTTON_LINE2	Button Linie 2
PA_VARBUTTON_NORMAL	Varbutton mit Attribut "normal"

Tabelle 15: Elementnamen in der Farbzordnungstabelle

Elementname	Beschreibung
PA_VARBUTTON_INVERS	Varbutton mit Attribut "invers"
PA_VARBUTTON_LOW	Varbutton mit Attribut "low"
PA_VARBUTTON_UNDERLINE	Varbutton mit Attribut "underline"
PA_VARBUTTON_BLINK	Varbutton mit Attribut "blink"
PA_VARBUTTON_GRAYED	Varbutton mit Attribut "grayed"
PA_VARBUTTON_BOLD	Varbutton mit Attribut "bold"
PA_VARBUTTON_INVISIBLE	Varbutton mit Attribut "invisible"
PA_VARBUTTON_LINE1	Varbutton Linie 1
PA_VARBUTTON_LINE2	Varbutton Linie 2
TOOLTIP	Tooltip

Damit nicht immer alle Farbwerte definiert werden müssen, gibt es einige Regeln, die die Farben zur Ersetzung definieren:

- Wenn keine Elemente für TABLE_... definiert werden, dann werden die Farben für Tabellen aus den FIELD_... Elementen gebildet.
- Wenn keine Elemente für SPECIAL_... definiert werden, dann werden die Farben für Special-Felder aus den FIELD_... Elementen gebildet.
- Wenn keine Elemente für PA_HEADER_... definiert werden, dann werden die Farben für Tabellenüberschriften aus den PA_TEXT_... Elementen gebildet.
- Wenn keine Elemente für PA_BUTTON_... definiert werden, dann werden die Farben für Buttons aus den PA_TEXT_... Elementen gebildet.
- Wenn keine Elemente für PA_VARBUTTON_... definiert werden, dann werden die Farben für Varbuttons aus den PA_BUTTON_... Elementen gebildet.

Ansonsten wird die Vordergrundfarbe eines nicht definierten Elements auf rot gesetzt und die Hintergrundfarbe auf grün. Tauchen diese beiden Farben im FIX/Win-Fenster auf, dann ist das ein Hinweis auf nicht definierte Elemente in der Farbzordnungstabelle.

Bei Elementen, die Linien besitzen, wird versucht diese dreidimensional zu zeichnen, indem die obere Linie in der Vordergrundfarbe des Linienelements und die untere Linie in der Hintergrundfarbe des Linienelements gezeichnet wird. Wenn dies nicht gewünscht wird, dann besteht die Möglichkeit, die beiden Begrenzungslinien in der gleichen Farbe zeichnen zu lassen, wie das Element selbst. Dazu sind die beiden Farbangaben der Linie auf die gleiche Farbe zu setzen, wie die Hintergrundfarbe des Elements beim Attribut "normal".

Beispiel:

Der folgende Ausschnitt definiert die Farben für Menüpunkte:

```
MENU_NORMAL    BLACK    WHITE
MENU_LOW       GRAY2    WHITE
MENU_INVERS    WHITE    BLUE1
MENU_UNDERLINE WHITE    BLUE1
MENU_LINE      WHITE    WHITE
```

Dabei wird für die Hintergrundfarbe von MENU_NORMAL die gleiche Farbe verwendet, wie für die Vorder- und Hintergrundfarbe von MENU_LINE. Das bewirkt, dass bei einem Menüpunkt mit dem Attribut "normal" für die obere und untere Begrenzungslinie die Farbe WHITE verwendet wird und für einen Menüpunkt mit dem Attribut "invers" die Farbe BLUE1. Diese Besonderheit in der Farbangabe bewirkt also, dass der für den kompletten Menüpunkt inclusive der Begrenzungslinien immer die Hintergrundfarbe verwendet wird, die zum aktuellen Videoattribut passt.

8 Anpassung von Schriftarten

Ab der Version 3.1.0 besteht die Möglichkeit, zur Darstellung von nicht proportionalen Zeichen, zusätzlich zu den mitgelieferten Schriftarten von FIX/Win Schriftarten aus dem Windowssystem zu verwenden. Gesteuert wird dies über Einstellungen im FIX/Win-Ressourcenformat (siehe [Allgemeines zu Ressourcen](#) auf Seite 79), die in der Datei

```
lookAndFeel.frc
```

einzutragen sind. Die Datei wird aus einem der Unterverzeichnisse

```
LookAndFeel/Fixwin-LAF
LookAndFeel/Fixwin-LAF-0
LookAndFeel/Fixwin-LAF-1
LookAndFeel/Fixwin-LAF-2
LookAndFeel/Fixwin-LAF-3
```

gelesen. Der Pfad der Datei wird nach dem Standardverfahren ermittelt (siehe [Verfahren zum Einlesen von Dateien](#) auf Seite 44).

8.1 Verwendung von Schriftarten des Windowssystems

Die Verwendung von Schriftarten des Windowssystems kann über folgende Einstellungen in `lookAndFeel.frc` erreicht werden:

Font

Diese Ressource legt den Namen der Schriftart fest, die für FIX/Win verwendet wird. Es kann jede im System verfügbare nicht proportionale Schriftart verwendet werden. FIX/Win prüft anhand der Maße bestimmter Zeichen, ob es sich um eine nicht proportionale Schriftart handelt.

Der Default für diese Ressource hängt vom Windows-Gebietsschema ab. In einer japanischen Umgebung wird die Schriftart *MS PGothic* verwendet. In allen anderen Umgebungen wird die Schriftart *FIXWIN* verwendet (siehe [Verwendung der mitgelieferten Schriftarten](#) auf Seite 71).

Beispiel

```
Font: Lucida Console
```

FontSize_M, FontSize_L, FontSize_H

Diese Ressourcen legen die Größen für die Schriftarten in den Bildgrößen Medium (`_M`), Large (`_L`) und Huge (`_H`) fest. Als Default werden die Werte 10, 16, 20 verwendet, wenn der entsprechende Wert nicht als Ressource definiert wird.

Beispiel

```
FontSize_M: 14
FontSize_L: 18
FontSize_H: 22
```

CellWidth_M, CellWidth_L, CellWidth_H

Legt die Breite einer Zeichenzelle fest.

Wenn der Wert -1 angegeben wird, dann wird die Breite anhand der Schriftart berechnet. Als Default werden die Werte 8, 10, 12 verwendet.

CellHeight_M, CellHeight_L, CellHeight_H

Legt die Höhe einer Zeichenzelle fest.

Wenn der Wert -1 angegeben wird, dann wird die Höhe anhand der Schriftart berechnet. Dazu wird das Zweifache von `CellSpace_[M|L|H]` zur Höhe der Schriftart addiert. Als Default werden die Werte 16, 20, 28 verwendet.

CellSpace_M, CellSpace_L, CellSpace_H

Legt den Platz über und unter der Schrift fest.

Einige Elemente (z.B. Felder) nutzen diesen Platz um Trennlinien zu zeichnen. Als Default werden die Werte 2, 2, 3 verwendet.

CellLineOffset_M, CellLineOffset_L, CellLineOffset_H

Legt den Abstand der Feldlinien vom Rand der Zelle fest.

Auf diesen beiden Positionen ($y + \text{CellLineOffset}$ und $y + \text{CellHeight} - \text{CellLineOffset}$) werden die inneren Linien gezeichnet. Für Felder werden zusätzlich zwei weitere Linien an den Positionen $y + \text{CellLineOffset} - 1$ und $y + \text{CellHeight} - \text{CellLineOffset} + 1$ gezeichnet.

Wichtiger Hinweis

Durch die Veränderung der Schriftart kann es zu einer Änderung der Größe einer Zeichenzelle kommen. Die Bitmaps für Tasten-Labels, Grafikzeichen und Schreibmarken sollten die gleiche Größe besitzen. Andernfalls werden sie skaliert, was in einigen Fällen zu unschönen Darstellungen führen kann.

Um die Größe einer Zeichenzelle zu bestimmen, kann der Menüpunkt

Diagnose/Zeicheninfo

aufgerufen werden. Danach ist ein Zeichen anzuklicken.

Da von der Änderung der Zellengröße immer alle Bitmaps einer Größe betroffen sind, sind vor der Anpassung alle Bitmaps aus dem Arbeitsverzeichnis von FIX/Win in das Gruppenverzeichnis zu kopieren. Bitmaps, für die bereits im Gruppenverzeichnis ein Ersatz vorhanden ist, sind dabei auszulassen. Danach ist die Größe von allen Bitmaps im Gruppenverzeichnis anzupassen. Die Bitmaps im Arbeitsverzeichnis von FIX/Win sollten nicht verändert werden, da sie einerseits von anderen Anwendungen genutzt werden können und andererseits bei der nächsten Installation wieder überschrieben werden.

8.1.1 Informationsdateien für Fullwidth-Zeichen

Bezüglich Fullwidth-Zeichen besteht das Problem, dass jede Plattform eine eigene Auffassung davon hat, ob ein Zeichen ein Fullwidth-Zeichen ist oder nicht. So liefert beispielsweise die Funktion `wcwidth()` bereits bei verschiedenen AIX Versionen unterschiedliche Ergebnisse.

Das führt dazu, dass es vorkommen kann, dass für Zeichen, die von der Schriftart her die volle Breite benötigen (2 Zeichenzellen), nur die halbe Breite reserviert wird (1 Zeichenzelle) und der Rest des Zeichens nicht sichtbar ist.

Lösungsansatz

Da letztendlich die Schriftart darüber entscheidet, ob ein Zeichen die volle Breite oder nur die halbe Breite benötigt, wird das Verhalten der Funktion `fx_wcwidth()` von der Schriftart abhängig gemacht.

Die Untersuchung von verschiedenen Schriftarten hat ergeben, dass es vorkommen kann, dass ein Zeichen je nach Schriftart unterschiedliche Breiten benötigt. Davon sind auch Zeichen im unteren Bereich von 0-255 betroffen. So benötigen beispielsweise die Zeichen β , \times , \pm , 1 , \sim in der Schriftart Courier die halbe Breite und in der Schriftart MS Gothic die volle Breite.

Da FIX und Windows nur die Zeichen im Bereich 0x0000-0xFFFF verwenden, wird aufgrund einer Schriftart eine Datei erstellt, die ein Array mit 65536 Bits enthält (8192 Bytes). Jedem Zeichen wird ein Bit zugeordnet. Wenn ein Bit gesetzt ist, dann bedeutet das, dass das Zeichen die volle Breite benötigt. Wenn das Bit nicht gesetzt ist, dann benötigt das Zeichen die halbe Breite, oder es benötigt gar keine Breite oder es ist nicht definiert. Um zu entscheiden, ob ein Zeichen die volle Breite benötigt, wird die Breite in Pixeln mit einem Referenz-Zeichen verglichen.

Die Datei wird von FIX und FIX/Win in den Speicher geladen und dient `fx_wcwidth()` zur Entscheidung, ob ein Zeichen die volle oder die halbe Breite benötigt. Da die Funktion `fx_wcwidth()` aufgrund von internen Tabellen entscheidet, ob ein Zeichen gar keine Breite (Rückgabewert 0) oder undefiniert ist, genügt die Information, ob ein Zeichen die volle oder die halbe Breite benötigt.

Erstellen der Fontinfo-Datei

Zur Erstellung der Datei ist das Programm

`FontInfo.exe`

zu verwenden. Es bekommt als Parameter den Namen der Schriftart zusammen mit dem Schalter

`/fontname <fontname>`

Zusätzlich ist ein weiterer Schalter notwendig, der angibt ob eine Fontinfo-Datei erzeugt werden soll oder ob weitere Informationen in eine Logdatei geschrieben werden sollen.

Der Schalter

`/create`

bewirkt, dass die Datei `config\<fontname>.ffi` erzeugt wird (Enthält <fontname> Leerzeichen, dann werden diese durch `_` ersetzt. Weiterhin wird die Groß-/Kleinschreibung dem tatsächlich installierten Font angepasst.). Sie enthält neben einem Dateikopf ein Array mit Informationen ob ein Zeichen die volle oder die halbe Breite benötigt.

Der Schalter

`/log`

bewirkt, dass die Datei `config\<fontname>.ffi` zum Test geladen wird und dass in die Logdatei `config\<fontname>.log` weitere Informationen zur Schriftart geschrieben werden. In der Logdatei sind folgende Informationen zu finden:

In der ersten Zeile steht der Name der Schriftart.

Fontinfo: MS Gothic

Dann wird die Breite des Referenz-Zeichens ausgegeben.

RefWidth=9

Jedes Zeichen, das diese Breite besitzt, gilt als Halfwidth-Zeichen. Zeichen, die die doppelte Breite besitzen, gelten als Fullwidth-Zeichen.

Danach folgt eine Liste mit allen Zeichen. Dabei wird der Code des Zeichens, das Zeichen selbst als UTF-8 Sequenz, die Zeichenklasse und die Breite in Pixeln ausgegeben. Um die Datei zu betrachten muss ein Programm verwendet werden, das UTF-8 beherrscht. Idealerweise stellt man bei diesem Programm die Schriftart ein, die auch der Logdatei zugrunde liegt. Bei den Zeichenklassen wird zwischen folgenden Werten unterschieden:

- UNDEFINED - Das Zeichen kommt in der Schriftart nicht vor.
- FULLWIDTH - Das Zeichen benötigt die volle Breite.
- HALFWIDTH - Das Zeichen benötigt die halbe Breite.
- NULLWIDTH - Das Zeichen benötigt keine Breite.
- OTHERWIDTH - Das Zeichen benötigt eine andere Breite, als voll oder halb.

Wenn das Zeichen nicht der Klasse UNDEFINED angehört, dann wird die Klasse mit dem verglichen, was `fx_wcwidth()` (aufgrund der geladenen Datei <fontname>.ffi) liefert. Wenn das Ergebnis sich unterscheidet, dann wird zusätzlich das Ergebnis von `fx_wcwidth()` ausgegeben. Da `fx_wcwidth()` anhand von internen Tabellen entscheidet, ob ein Zeichen die Breite 0 hat, kann es in diesem Fall zu Unterschieden kommen. Da es sich nur um wenige Zeichen handelt (bei MS Gothic sind es 5) sind diese Unterschiede zu vernachlässigen.

Beispiel

Char: 00b1 : ± : FULLWIDTH : 18

Das Zeichen '±' mit dem Code 001b benötigt die volle Breite und ist 18 Pixel breit.

Wenn bei der Liste der Zeichen die Zeichenklasse wechselt, dann wird eine Zusammenfassung ausgegeben, die den Bereich der Zeichen angibt.

Beispiel

Range: 00b0 - 00b1 : FULLWIDTH : cnt=2

Das nächste Zeichen ist kein Zeichen mit voller Breite. Die beiden vorherigen Zeichen (cnt=2) belegen den Bereich 00b0 - 00b1 und sind Fullwidth-Zeichen.

Verwendung in FIX/Win

Zur Verwendung in FIX/Win sind keine weiteren Schritte zu unternehmen. FIX/Win liest die Fontinfo-Datei aus dem Verzeichnis in dem sie FontInfo.exe ablegt und verwendet ebenfalls den Namen der Schriftart als Dateiname.

Zusätzlich wird FIX/Win zusammen mit einigen wichtigen Fontinfo-Dateien ausgeliefert, so dass für diese Schriften die Erstellung einer Datei entfallen kann. Es kann jedoch vorkommen, dass auf einem System eine andere Version einer Schriftart installiert ist. In diesem Fall ist die entsprechende Fontinfo-Datei neu zu erstellen.

Abweichend davon ist es auch möglich, die Fontinfo-Datei im Unterverzeichnis `config` im Gruppenverzeichnis abzulegen. Wird die Datei dort gefunden, dann wird diese verwendet und die Datei im Arbeitsverzeichnis von FIX/Win bleibt unbeachtet. (gleiches Verfahren, wie bei allen anderen Dateien in `config`)

Verwendung in der Benutzerbibliothek

Da in der Benutzerbibliothek die Ermittlung der Zeichenbreite nützlich ist, bietet der Kern von FIX/Win eine Funktion dazu an.

Definition

```
int WcsWidth(void* p_callID, wchar_t* wcs, int n)
```

Makros

```
FW_WCSWIDTH, FWFCT_WCSWIDTH(callback, varname)
```

Beschreibung

Die Funktion ermittelt die Breite der Zeichen in `wcs`. Der Parameter `n` enthält die maximale Anzahl Zeichen. Ist `n` kleiner als die Länge der Zeichenkette in `wcs`, dann werden nur `n` Zeichen berücksichtigt. Andernfalls werden alle Zeichen berücksichtigt.

Da die Fontinfo-Datei erst beim Laden des LookAndFeels gelesen wird, kann die Zeichenbreite vorher nicht ermittelt werden. In diesem Fall liefert die Funktion den Wert -2 zurück.

Beispiel

Das folgende Beispiel gibt die Breite jedes Textes einer Paintarea aus.

```
WcsWidthFct WcsWidth = NULL;
```

```
MsgWinShowMsgFctMsgWinShowMsg = NULL;
```

```
FWOWN_API _TCHAR* CALLBACK FwownDrawPaintArea(  
    CallBackFct CallBack, void* p_callID,  
    HDC p_hDCPA, int p_width, int p_height,  
    unsigned long p_pa_id, short p_pa_type,  
    _TCHAR* p_text, unsigned short p_pa_attr, short p_pa_align,
```

```

long p_longval1, long p_longval2,
long p_longval3, long p_longval4,
_TCHAR* p_ms_name, short p_objclass,
int p_size, int p_style) {

    int px;
    _TCHAR buffer[255];

    FWFCT_WCSWIDTH(CallBack, WcsWidth);
    FWFCT_MSGWINSHOWMSG(CallBack, MsgWinShowMsg);

    px = WcsWidth(p_callID, p_text, 1000);
    _tsprint_s(buffer, _T("WcsWidth(%s)=%d

    MsgWinShowMsg(p_callID, buffer);

    ...
}

```

8.2 Verwendung der mitgelieferten Schriftarten

Zur Verwendung der mitgelieferten Schriftarten (oder von Schriftarten, die aus diesen hervorgehen und angepasst wurden) ist der Wert

Font: FIXWIN

in der Datei `lookAndFeel.frc` zu setzen. Da der Wert FIXWIN der Default für die Ressource Font ist, kann der Eintrag auch komplett wegelassen werden.

FIX/Win verwendet dann Schriftarten im Windows-.FNT Format. Dabei handelt es sich um rasterorientierte Schriften, die in einer bestimmten Größe und einem bestimmten Zeichensatz bereitzustellen sind. Sowohl die Größe, als auch der Zeichensatz wird im Dateinamen kodiert, so dass FIX/Win erkennen kann, welche FNT-Datei zu laden ist. Beim Laden wird das im Abschnitt [Aufbau eines Gruppenverzeichnisses](#) auf Seite 41 beschriebene Suchverfahren verwendet. Je nach Dateiname und Zeichensatz wird folgende Datei geladen:

Tabelle 16: Namen der Schriftdateien

Zeichensatz	Medium	Large	Huge
ISO-15, GERMAN7 IBM437 UTF-8	FW-M-L15.FNT	FW-L-L15.FNT	FW-H-L15.FNT
ISO-2	FW-M-L2.FNT	FW-L-L2.FNT	FW-H-L2.FNT

Der in der Tabelle angegebene Zeichensatz bezieht sich auf FIX und wird über die Umgebungsvariable `FXCHARSET` für die Anwendung eingestellt. Beim Senden von Daten an FIX/Win werden diese in den Unicode-Zeichensatz konvertiert. Eine Konvertierung dieses Zeichensatzes in den Zeichensatz der Schriftart (Windows-1252 für (*-L15) und Windows-1250 für (*L2)) wird von Windows unmittelbar vor dem Zeichnen der Buchstaben vorgenommen.¹

Bei der Anpassung der Schriften durch einen Fonteditor sind zwei Dinge zu beachten:

- Damit die Schriften zu den Semigrafikzeichen und den Bildgrößen von FIX/Win passen, sind folgende Größen zu verwenden:

1. Beispielsweise besitzt das Euro-Zeichen den Code 8364 im Unicode-Zeichenbereich. Intern steht es auch mit diesem Code im Speicher von FIX/Win (Nachzuprüfen über die Funktion `Diagnose/Zeicheninfo`). Bei der Ausgabe wird jedoch das Zeichen mit dem Code 0x80 verwendet, wenn die Schriftart `FW-[M|L|H]-L15.fnt` verwendet wird.

Tabelle 17:

Größe	Breite	Höhe
Medium	8	12
Large	10	16
Huge	12	22

- Damit die Umsetzung von Unicode in den Zeichensatz der Schriftart funktioniert, müssen die Codes der Zeichen stimmen und der Wert für das Attribut Charset ist richtig zu setzen:

Tabelle 18:

Datei	Wert	Bezeichnung
FW-[M L H]-L15.fnt	0	ANSI_CHARSET
FW-[M L H]-L2.fnt	238	EASTEUROPE_CHARSET

8.3 Definition weiterer Schriftarten

Zusätzlich zu der Schriftart, die für den Text des FIX/Win Fensters verwendet werden, verwendet FIX/Win weitere Schriftarten, die ebenfalls in der Datei `lookAndFeel.frc` eingestellt werden können.

PropFont

Legt den Namen der proportionalen Schriftart für Paintareas fest.

Der Default für diese Ressource hängt vom Windows-Gebietsschema ab. In einer japanischen Umgebung wird die Schriftart *MS PGothic* verwendet. In allen anderen Umgebungen wird die Schriftart *Arial* verwendet.

Beispiel

PropFont: Arial

PropFontSize_M, PropFontSize_L, PropFontSize_H

Diese Ressourcen legen die Größen für die proportionalen Schriftarten in den Bildgrößen Medium (*_M*), Large (*_L*) und Huge (*_H*) fest. Als Default werden die Werte 15, 17, 20 verwendet, wenn der entsprechende Wert nicht als Ressource definiert wird.

IMEFontSize_M, IMEFontSize_L, IMEFontSize_H

Legt die Größe der Schriftart für das IME-Compositionwindow fest. Als Schriftart wird die durch die Ressource `Font` definierte Schriftart verwendet. Wird als Größe `-1` angegeben, dann wird die Größe aus einer der Ressourcen `FontSize_M`, `FontSize_L`, `FontSize_H` gelesen.

TooltipFont

Legt den Namen der Schriftart für Tooltips fest.

Der Default für diese Ressource hängt vom Windows-Gebietsschema ab. In einer japanischen Umgebung wird die Schriftart *MS PGothic* verwendet. In allen anderen Umgebungen wird die Schriftart *Arial* verwendet.

TooltipFontSize_M, TooltipFontSize_L, TooltipFontSize_H

Legt die Größe der Schriftart für Tooltips in jeder der drei Bildgrößen fest.

9 Anpassung der Schreibmarke

Das Aussehen der Schreibmarke (Caret, Cursor) wird in FIX/Win durch eine zweifarbige Bitmap bestimmt. Beim Laden wird das im Abschnitt *Aufbau eines Gruppenverzeichnis* auf Seite 41 beschriebene Suchverfahren verwendet. Der Dateiname der Bitmap und die Ausmaße in Pixeln hängen von der aktuellen Bildgröße ab und sind folgender Tabelle zu entnehmen, wenn die mitgelieferten Schriftarten verwendet werden:

Tabelle 19: Caret Bitmaps

Größe	Dateiname	Breite	Höhe
Medium	caret-M.bmp	8	16
Large	caret-L.bmp	10	20
Huge	caret-H.bmp	12	28

Bei der Verwendung von Schriftarten aus dem Windowssystem ergibt sich die Größe aus der Größe der Zelle. Die Größe der Zelle kann über den Menüpunkt *Diagnose/Zeicheninfo* ermittelt werden.

Neben der Bitmap für die Schreibmarke der entsprechenden Bildgröße, kann eine zweite Bitmap in doppelter Breite angelegt werden. Diese Bitmap wird von FIX/Win verwendet, wenn sich die Schreibmarke über einem Fullwidth-Zeichen befindet. Der Name muss die Form:

caret-<n>-full.bmp

haben. <n> ist der Buchstabe, der die Bildgröße bezeichnet (M(edium), L(arge) oder H(uge)). Existiert eine solche Datei nicht, dann wird für Fullwidth-Zeichen die gleiche Schreibmarke verwendet, wie für Halfwidth-Zeichen.

10 Anpassen von Meldungen, Menüpunkten und Dialogen

Meldungen, Menüpunkte und Dialoge werden von FIX/Win - wie unter Windows üblich - aus Ressourcen geladen. Die Ressourcen von FIX/Win stehen in zwei DLL-Dateien:

ResFixwin.dll - enthält Ressourcen für das Standard-Framework.

ResFixwinCore.dll - enthält Ressourcen der Kern-Klassen.

Durch die Angabe der Kommandozeilenschalter `/resfw <Datei>` (für das Standard-Framework) und `/resfwc <Datei>` (für die Kern-Klassen) ist es möglich, andere Dateien beim Start von FIX/Win zu laden. Damit sind Anpassungen an eigene Bedürfnisse (z.B. das Weglassen von Menüpunkten) oder an andere Sprachen möglich. Als Vorlage für eine eigene DLL kann der Quellcode der Originaldateien verwendet werden, der in den Unterverzeichnissen

src/ResFixwin

src/ResfixwinCore

zu finden ist.

Ab der Version 3.1.0 besitzt FIX/Win einen überarbeiteten Mechanismus zum Laden der Ressourcen. Dieser versucht zuerst die Ressourcen zu laden, die zum eingestellten Gebietschema passen. Werden diese nicht gefunden, dann werden die Ressourcen geladen, die für die neutrale Sprache definiert wurden (Deutsch). Damit ist es möglich, eine DLL zu erstellen, die Ressourcen für mehrere Sprachen enthält und sich an das jeweilige Windowssystem anpaßt. Die ausgelieferten Quellen enthalten als Beispiel Ressourcen für Japanisch.

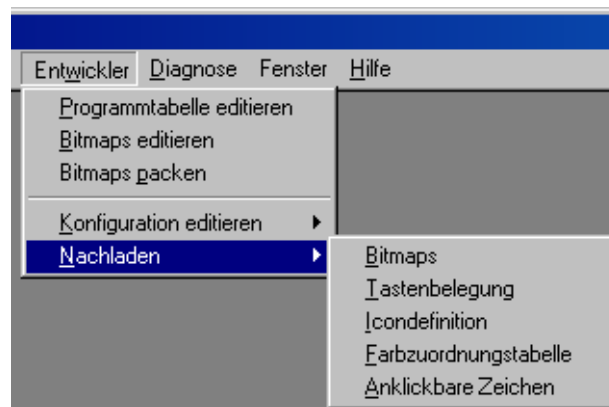
11 Bedienung durch das Entwicklermenü

Wird FIX/Win mit dem Kommandozeilenparameter `/dev` gestartet, dann können alle Einstellungen über das Menü "Entwickler" vorgenommen und nachgeladen werden, ohne dass FIX/Win verlassen werden muss. Zum Bearbeiten der Einstellungen sind allerdings zwei externe Programme notwendig:

- Ein *Texteditor*, mit dem die Konfigurationsdateien verändert werden können.
Am besten wird hier der Standard-Windows-Editor `notepad.exe` verwendet. Er ist in jedem Windows-Paket enthalten und reicht voll und ganz aus, um die Konfigurationsdateien zu bearbeiten. Damit FIX/Win diesen Editor verwendet, ist die Ressource `FwEditTool` zu setzen:
`FwEditTool: notepad.exe`¹
Außerdem sollte sich `notepad.exe` im Pfad befinden.
Bei Verwendung eines anderen Texteditors sollte darauf geachtet werden, dass dieser seine Texte im ANSI-Zeichensatz ablegt, so dass Umlaute und Sonderzeichen korrekt übernommen werden.
- Ein *Bitmapeditor*, mit dem die Bitmaps von FIX/Win verändert werden können.
Analog zum Texteditor kann hier das Standard-Windows-Programm `pbrush.exe` verwendet werden. Da dieses Programm jedoch eher auf größere Bitmaps angelegt ist, sollte besser ein anderes Tool verwendet werden. Um FIX/Win den Bitmapeditor bekannt zu machen, ist die Ressource `FwBmpTool` zu setzen:
`FwBmpTool:pbrush.exe`²

11.1 Menüpunkte des Entwicklermenüs

Abb. 12 FIX/Win-Entwicklermenü



Im folgenden werden die Menüpunkte des Entwicklermenüs kurz beschrieben:

Menüpunkt "Programmtabelle editieren"

Der Editor `edprgtab.exe` zur Bearbeitung der Programmtabelle wird aufgerufen. Da die Programmtabelle vor jedem Verbindungsaufbau neu geladen wird, ist ein explizites Nachladen nach dem Abspeichern der geänderten Tabelle nicht notwendig. Zum Aufbau der Datei selbst wird auf [Eintragen des Programmes auf Seite 23](#) verwiesen.

Menüpunkt "Bitmaps editieren"

Hiermit wird in den *Edit Bitmap*-Modus umgeschaltet. Der Mauszeiger ändert sich dabei in ein Edit-Symbol (siehe [Mauszeiger unter FIX/Win](#) auf Seite 29), mit dem auf ein Semigrafikzeichen des FIX-Bildschirms oder auf ein Bitmap der IconBox geklickt werden kann. Darauf hin testet FIX/Win ab, ob ein entsprechendes Bitmap im Gruppenverzeich-

1. bei Verwendung eines anderen Editors der entsprechende Name

nis existiert. Ist dies nicht der Fall, so kopiert FIX/Win ein Standardbitmap aus dem Arbeitsverzeichnis von FIX/Win als Vorlage in das entsprechende Unterverzeichnis des Gruppenverzeichnisses, bei Semigrafikzeichen abhängig von Bildschirmgröße und -stil. Nach Abspeichern des geänderten Bitmaps und erneutem Packen durch das Programm `bmpack` können die Auswirkungen sichtbar gemacht werden, indem einer der folgenden Menüpunkte aufgerufen wird:

- *Nachladen/Icondefinition* wenn ein Bitmap der IconBox geändert wurde
- *Nachladen/Bitmaps* wenn ein Bitmap für ein Semigrafikzeichen geändert wurde

Sollte der Menüpunkt zufällig angeklickt worden sein, kann der *Edit Bitmap*-Modus durch nochmalige Anwahl des Menüpunktes ausgeschaltet werden. Zusätzlich zeigt ein Haken am Menüpunkt an, ob der *Edit Bitmap*-Modus aktiv ist. Mehr über Bitmaps für Semigrafikzeichen ist in [Verwendung von Semigrafikzeichen](#) auf Seite 57 zu erfahren.

Menüpunkt "Bitmaps packen"

Mit diesem Menüpunkt kann das Programm `bmpack` aufgerufen werden. Durch Angabe von Kommandozeilenoptionen über die Ressource `BMPACKOPTIONS` kann ihm ein bestimmtes Gruppenverzeichnis und eine bestimmte Konfiguration mitgegeben werden. Näheres über das Packen von Bitmaps ist im Abschnitt [Packen von Bitmaps](#) auf Seite 60 zu finden.

Menüpunkte "Konfiguration editieren" und "Nachladen"

Der erste Menüpunkt bietet verschiedene Untermenüpunkte zur Bearbeitung der Konfigurationsdateien von FIX/Win an. Hierzu wird der Texteditor zum Editieren der entsprechenden Konfigurationsdatei aufgerufen. Diese wird abhängig von der laufenden Anwendung aus dem entsprechenden Gruppenverzeichnis gelesen. Sollte eine Datei nicht vorhanden sein, so bietet FIX/Win an, eine Standarddatei als Vorlage in das Gruppenverzeichnis zu kopieren. Das Nachladen der Konfigurationsdateien erfolgt über einen Untermenüpunkt des Menüs *Nachladen*, welcher den gleichen Namen hat wie der entsprechende Untermenüpunkt des Menüs *Konfiguration editieren*.

Der Aufbau der Dateien selbst kann den folgenden Abschnitten entnommen werden:

- *Tastenbelegung* [Abschnitt 4 auf Seite 53](#)
- *Icondefinition* [Abschnitt 3 auf Seite 45](#)
- *Farbzuordnungstabelle* [Abschnitt 7 auf Seite 63](#)

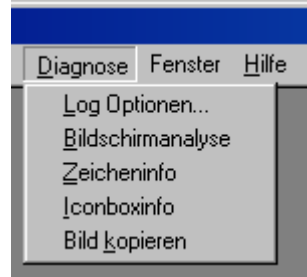
12 Möglichkeiten des Diagnosemenüs

Zur Aktivierung dieses Menüs ist beim Start von FIX/Win der Kommandozeilenparameter

```
/diag <nr>
```

anzugeben. Eine Beschreibung des Wertes <nr> folgt im nächsten Abschnitt.

Abb. 13 FIX/Win-Diagnosemenü



Menüpunkt "Logoptionen ..."

Mit Hilfe dieses Menüpunktes kann die Debugausgabe konfiguriert werden. Die Ausgaben werden in die Datei `fixwin.log` geschrieben. Zusätzlich erfolgt eine Ausgabe über die Windows-Funktion `OutputDebugString()`, so dass die Ausgaben während des Laufs von FIX/Win mit entsprechenden Tools (z.B. DebugView von sysinternals) beobachtet werden können. Jede Ausgabe wird einer von mehreren Gruppen zugeordnet. Zu Beginn jeder Zeile wird der Gruppenname ausgegeben. Damit besteht die Möglichkeit, die Ausgaben einer Gruppe zu filtern (z.B. mit `grep`). Jeder Gruppe wird ein Wert aus `include/FixwinCore.h` zugeordnet (Makros, die mit LOG beginnen). Aus den gewünschten Gruppen ist durch oder-Verknüpfung ein Wert zu bilden, der dem Kommandozeilen Parameter `/diag` mitgegeben wird. Alle Gruppen, die durch den Bitvector definiert werden, werden zu Beginn eingeschaltet. Um die Gruppen zur Laufzeit umzuschalten, kann über diesen Menüpunkt ein Dialog zur Konfiguration gestartet werden, in dem jede Gruppe an- oder ausgeschaltet werden kann.

Die Debugausgaben von FIX/Win sind nicht dokumentiert. Sie können sich bei Bedarf mit jeder Auslieferung von FIX/Win ändern. Sie dienen in erster Linie NSI, um Fehler verfolgen zu können, die auf einem Rechner des Kunden auftreten.

Menüpunkt "Bildschirmanalyse"

Durch das Auswählen dieses Menüpunktes wird der Inhalt des aktuellen FIX/Win-Fenster in die Datei `fixwin.log` geschrieben. Zu jeder Zeichenzelle existieren verschiedene Informationen wie z.B. das Zeichen selbst, das Attribut, das FIX-Element (Feld, Menüpunkt, Selospalte, ...) zu dem das Zeichen gehört. Für jede dieser Informationen wird eine Ausgabe in Form einer Tabelle erzeugt, in der der Inhalt einer Zelle durch ein Zeichen darstellt wird. Eine Legende erläutert den Inhalt einer Zelle.

Die Ausgabe kann verwendet werden, um Fehlern in der Darstellung auf die Spur zu kommen.

Menüpunkt "Zeicheninfo"

Bei Aufruf dieses Menüpunktes wechselt der Mauszeiger in eine Lupe, mit der das zu analysierende Zeichen angeklickt werden kann (siehe [Mauszeiger unter FIX/Win](#) auf Seite 29). Analog dem Menüpunkt *Bildschirmanalyse* werden die Informationen für ein einzelnes Zeichen auf dem Bildschirm ausgegeben. Die gelieferten Daten können wichtig für das Bearbeiten der Konfigurationsdateien sein.

Menüpunkt "IconBoxinfo"

Durch Aufruf dieses Menüpunktes liefert FIX/Win Informationen über die aktuellen Einstellungen der IconBox:

- ID der aktuellen Iconliste
- ID des FIX-Iconsets

- ID des User-Iconsets

Diese Informationen werden oft beim Bearbeiten der Icondefinitionsdatei benötigt.

Menüpunkt "Bild kopieren"

Mittels des Untermenüpunktes "als Grafik" wird der momentane Inhalt des FIX/Win Fensters in der Zwischenablage als Bitmap-Grafik abgelegt. Diese Grafik kann so als Hardcopy in ein entsprechendes Textprogramm eingefügt werden.

13 Verwendung von FIX/Win-Ressourcen

In der Dokumentation zu FIX/Win ist der Begriff "Ressourcen" doppelt belegt. Zum einen sind damit Windows-Ressourcen gemeint. Diese Ressourcen dienen unter Windows zum Erstellen von Menüs, Dialogen und sprachabhängigen Texten. Sie spielen bei der Umsetzung von FIX/Win in eine andere Sprache eine Rolle und beim Erweitern von FIX/Win über die Programmierschnittstelle. Zum anderen wird der Begriff "Ressourcen" für Einstellungen verwendet, die in externen Dateien gehalten werden. Dieser Abschnitt bezieht sich auf diese Art der Ressourcen. Sie werden deshalb hier kurz als Ressourcen bezeichnet. In anderen Abschnitten und Dokumentationen werden sie zur Unterscheidung von den Windows-Ressourcen als FIX/Win-Ressourcen bezeichnet.

13.1 Allgemeines zu Ressourcen

Für die Ressourcen von FIX/Win wird das gleiche Format verwendet, wie für FIX und REP. Das in diesem Abschnitt dokumentierte gilt somit auch für diese Produkte. Allerdings wird unter FIX/Win die Dateierdung `.frc` (statt `.rc`) für Ressourcendateien verwendet, da die Endung `.rc` bereits für die Windows-Ressourcen verwendet wird.

Aufbau der Ressourcendatei

Eine Ressourcendatei besteht aus mehreren Textzeilen. Eine Zeile kann dabei aus folgendem bestehen:

- eine Zuordnung von Ressourcenname zu Ressourcwert mit optionalem Kommentar am Ende:
`<Ressourcenname>:<Wert> !Kommentar`
- eine Kommentarzeile oder Leerzeile:
`! Kommentar`
- einer Anweisung zum Einschließen einer anderen Ressourcendatei:
`#include <Datei>`

Aufbau der Ressourcezuweisung

Die Ressourcezuweisung besteht aus einem Ressourcenamen und einem Ressourcwert. Der Ressourcenname besteht aus einem Identifier (Zeichenfolge von Buchstaben, Zahlen und Unterstrich, die nicht mit einer Zahl beginnt) und einem optionalen Programmnamen am Anfang, dem ein Punkt folgt. Dieser optionale Programmname wird von FIX/Win jedoch nicht unterstützt.

Beispiele:

<code>Ressourcel</code>	- gültiger Ressourcenname
<code>5te_Ressource</code>	- ungültiger Ressourcenname
<code>Prog1.Res1</code>	- Ressource <code>Res1</code> für Programm <code>Prog1</code>

Der Ressourcwert besteht aus beliebigen Zeichen, wobei das `$`-Zeichen jedoch eine Sonderbedeutung hat. Weiterhin werden führende und folgende Leerzeichen entfernt.

Zugriff auf andere Ressourcwerte

Mittels des Konstrukts '`$(<Ressourcwert>)`' kann auf andere Ressourcwerte zugegriffen werden. Dieser Ressourcwert muss jedoch in der Ressourcendatei vor dem Zugriff definiert sein. Beispiel:

Hauptverzeichnis:	<code>/home/user</code>
Unterverzeichnis:	<code>\$(Hauptverzeichnis)/subdir</code>

Die Ressource `Unterverzeichnis` erhält den Wert `/home/user/subdir`.

Maximalwerte

Für die Ressourcdatei sind folgende Maximallängen definiert:

- Länge einer Zeile: 512
- Länge eines Ressourcenamens: 255
- Länge eines Ressourcwertes nach der Ersetzung anderer Ressourcwerte einschließlich aller Zeichen zur Ersatzdarstellung: 255
- Länge einer Pfadangabe: 255

Einschließen von anderen Ressourcdateien

Mittels der Anweisung

```
#include <Datei>
```

kann eine weitere Ressourcdatei eingeschlossen werden. Der Wert `<Datei>` muss einen gültigen Dateinamen enthalten. Er darf aber auch Zugriffe auf Ressourcwerte enthalten.

Beispiel:

```
HOME:      /user/home  
#include $(HOME)/my.rc
```

liest die Datei `/home/user/my.rc` und setzt die dort angegebenen Ressourcen.

13.2 Ressourcdateien von FIX/Win

FIX/Win unterscheidet grundsätzlich fünf Arten von Ressourcdateien:

- Ressourcdatei mit globalen Einstellungen
- Ressourcdatei mit Einstellungen für das LookAndFeel
- Ressourcdatei mit Einstellungen für die IconBox
- Ressourcdatei mit anwendungsunabhängigen Ressourcen
- Ressourcdatei mit anwendungsabhängigen Ressourcen

Die globalen Einstellungen stehen in der Datei `config/fixwin.frc` im Arbeitsverzeichnis von FIX/Win. Da diese Datei bereits beim Start von FIX/Win gelesen wird, besteht keine Möglichkeit, die Datei aus dem Gruppenverzeichnis zu lesen. Dieses Verzeichnis wird erst beim Verbindungsaufbau ermittelt. Die möglichen Einstellungen sind im Abschnitt [Globale Einstellungen von FIX/Win auf Seite 21](#) zu finden.

Der Aufbau der Ressourcdatei für das LookAndFeel wird im Abschnitt [Anpassung von Schriftarten](#) auf Seite 67 beschrieben. Ressourcdateien für die IconBox werden im Abschnitt [Einstellungen über Ressourcen](#) auf Seite 50 beschrieben.

Der Name der Ressourcdatei mit anwendungsabhängigen Ressourcen ist in der Programmtabelle vermerkt. Der Name der anwendungsunabhängigen Ressourcen lautet immer `'fw_usr.frc'`. Die folgenden Abschnitte beziehen sich nur auf diese beiden Dateien.

Lesen der Ressourcdatei

Beim Verbindungsaufbau werden die Ressourcen gelesen und ausgewertet. Dabei wird zuerst die Datei mit anwendungsunabhängigen Ressourcen gelesen und dann die Datei mit anwendungsabhängigen Ressourcen. Zum Auffinden der Dateien wird folgendes Schema verwendet:

- Wurde FIX/Win ein Benutzerverzeichnis mitgeteilt (Parameter `/userdir`), dann wird versucht die Ressourcdatei aus diesem Verzeichnis zu lesen.
- Ist dort die Datei nicht vorhanden, oder wurde kein Benutzerverzeichnis angegeben, dann wird versucht, die Ressourcdatei aus dem Unterverzeichnis `config` des Gruppenverzeichnisses zu lesen.
- Wird die Ressourcdatei auch hier nicht gefunden, dann wird eine Defaultressourcdatei aus dem Unterverzeichnis `config` des FIX/Win Arbeitsverzeichnisses gelesen. Der Name der Ressourcdatei mit den anwendungsabhängigen Ressourcen wird dann nicht mehr aus der Programmtabelle ermittelt, sondern lautet `'fw_prg.frc'`.

Schreiben von Ressourcwerten

Mittels des Menüpunktes 'Optionen/Speichern' werden die aktuellen Einstellungen in den Ressourcdateien abgelegt.

- Beim Speichern von Ressourcen wird die alte Ressourcdatei gelesen und nach den Zeilen mit dem Inhalt `'!F_GENRES!'` durchsucht. Die folgenden Zeilen bis zum erneuten Auftauchen der Zeile mit dem Inhalt `'!F_GENRES!'` werden entfernt und durch Ressourcen, die die aktuelle Einstellung beschreiben ersetzt.
- Der Pfad der alten Ressourcdatei wird nach dem gleichen Verfahren ermittelt, wie der der Ressourcdatei beim Einlesen nach dem Verbindungsaufbau.
- Der Pfad der neuen Ressourcdatei wird nach folgendem Schema ermittelt:
 - wurde ein Benutzerverzeichnis angegeben, dann wird die neue Ressourcdatei in diesem Verzeichnis abgelegt.
 - ansonsten wird die Datei im Gruppenverzeichnis angelegt.

Wenn die Einstellungen schon einmal gespeichert wurden, dann handelt es sich bei der alten und der neuen Ressourcdatei um die gleiche Datei. In diesem Fall wird die Endung der neuen Datei auf `'.tmp'` gesetzt. Nach dem Erzeugen der neuen Datei wird die alte Datei gelöscht und die neue Datei auf den Namen der alten umbenannt. Wenn die Einstellungen das erste Mal gespeichert werden, dann wird anhand der Dateien im Arbeitsverzeichnis von FIX/Win eine neue Datei im Gruppenverzeichnis oder im Benutzerverzeichnis angelegt.

Ein Beispiel zum Verständnis:

Eine neu erstellte Anwendung soll mit FIX/Win bedient werden. Es existiert noch keine Ressourcdatei. Als Ressourcdatei wird in der Programmtabelle `'myapp.frc'` angegeben. Für die Anwendung wird ein leeres Gruppenverzeichnis erzeugt. Beim Starten der Anwendung werden die Ressourcen aus den Dateien `'fw_usr.frc'` und `'fw_prg.frc'` aus dem FIX/Win Arbeitsverzeichnis gelesen, da im Gruppenverzeichnis keine Ressourcdateien vorhanden sind. Beim Speichern werden diese Dateien als Vorlage verwendet und die neuen Ressourcdateien werden im Gruppenverzeichnis abgelegt. Der Dateiname der anwendungsabhängigen Ressourcen lautet dabei `'myapp.frc'`, da er aus der Programmtabelle gelesen wird.

Sollen nun mehrere Benutzer mit der Anwendung arbeiten und alle sollen das gleiche (im Netz installierte) Gruppenverzeichnis verwenden, dann müssen diese beim Start von FIX/Win ein Benutzerverzeichnis angeben (Schalter `/userdir`). Beim ersten Start werden die Ressourcdateien jedoch aus dem Gruppenverzeichnis der Anwendung gelesen, weil diese im Benutzerverzeichnis noch nicht vorhanden sind. Diese Dateien werden auch beim Speichern als Vorlage verwendet. Die neuen Ressourcdateien werden jedoch im jeweiligen Benutzerverzeichnis für jeden Benutzer getrennt mit seinen jeweiligen Einstellungen abgelegt.

Jede Anwendung hat also die gleiche Vorlage aus dem Arbeitsverzeichnis von FIX/Win und jeder Benutzer hat die gleiche Vorlage aus dem Gruppenverzeichnis der Anwendung. Jedoch beim Speichern (und erneutem Einlesen) hat jeder Benutzer (jede Anwendung) sein eigenes Verzeichnis und somit seine eigenen Ressourcen.

Die Unterteilung der Ressourcen in anwendungsabhängige und anwendungsunabhängige kommt erst dann zum Tragen, wenn mit mehreren Anwendungen gearbeitet wird. Die anwendungsunabhängigen Ressourcen sind nämlich für alle Anwendungen gültig.

Einschließen von Dateien

Die Syntax von Ressourcdateien erlaubt es Dateien einzuschließen. Dies kann in der folgenden Weise genutzt werden:

Die Ressourcdatei `'fw_usr.frc'` im Gruppenverzeichnis enthält den Eintrag:

```
#include $(GrpDir)/deflt.frc
```

Die Datei 'deflt.frc' im Gruppenverzeichnis wird gelesen und die dort definierten Ressourcen werden gesetzt. Werden nun die Einstellungen gespeichert, dann wird im Benutzerverzeichnis die Datei 'fw_usr.frc' angelegt, die ebenfalls die 'include' Anweisung enthält. So besteht die Möglichkeit, durch Änderung der Datei 'deflt.frc' Ressourcen für alle Benutzer zu ändern, ohne alle Ressourcendateien in sämtlichen Benutzerverzeichnissen ändern zu müssen. Dies funktioniert jedoch nicht, wenn Ressourcen nach der 'include' Anweisung durch erneute Definition überschrieben werden.

Einlesereihenfolge

Es werden zuerst die anwendungsunabhängigen Ressourcen eingelesen und dann die anwendungsabhängigen. Dabei gilt, dass Definitionen von bereits definierten Ressourcen diese überschreiben. Wie man sich dies zunutze machen kann, zeigt das folgende

Beispiel:

Benutzer U arbeitet vorzugsweise im Bildstil 3. Deshalb hat er diesen durch 'Optionen/Speichern' in der Ressourcendatei 'fw_usr.frc' abgespeichert. Nun wird eine neue Anwendung programmiert, welche selbst definierte Bitmaps enthält. Diese Bitmaps liegen aber nur für den Bildstil 2 vor. Um nun die Angabe des Benutzers U zu überschreiben wird eine Datei 'inc.frc' im Gruppenverzeichnis angelegt und von der anwendungsabhängigen Ressourcendatei 'newapp.frc' eingeschlossen. Die Datei 'inc.frc' enthält die Definition:

```
ScreenStyle: 2
```

Startet Benutzer U nun die neue Anwendung mit FIX/Win, dann wird als erstes die anwendungsunabhängige Ressourcendatei aus seinem Benutzerverzeichnis gelesen, welche die Ressource `ScreenStyle` auf 3 setzt. Danach wird die anwendungsabhängige Ressourcendatei `newapp.frc` gelesen, welche `inc.frc` einschließt und die Ressource `ScreenStyle` auf den Wert 2 umsetzt. Speichert der Benutzer seine Ressourcen, dann wird `newapp.frc` aus dem Gruppenverzeichnis als Vorlagedatei verwendet und die Datei `newapp.frc` für diese Anwendung im Benutzerverzeichnis angelegt. Da diese Datei ebenfalls die `include` Anweisung enthält, ist es später leicht möglich, durch Entfernen der obigen Zeile aus der Datei `inc.frc`, die Einschränkung auf den Bildstil 2 wieder aufzuheben.

13.3 In FIX/Win verwendete Ressourcen

Folgende Ressourcen können in FIX/Win verwendet werden:

generiert bei Menüpunkt Optionen/Speichern

anwendungsabhängig:

Ressource	Beschreibung	Werte
<code>FixWindow_x</code>	X-Position des FIX/Win Fensters	integer
<code>FixWindow_y</code>	Y-Position des FIX/Win Fensters	integer

anwendungsunabhängig:

Ressource	Beschreibung	Werte
<code>Viewer</code>	Dateibetrachter	String
<code>Viewerfile</code>	Ausgabedatei	String
<code>ScreenSize</code>	Bildgröße	1,2,3
<code>ScreenStyle</code>	Bildstil	0,1,2,3

Ressource	Beschreibung	Werte
IconboxVisible	Iconbox angeschaltet	TRUE/FALSE
IconboxWide	Iconbox horizontal oder Vertikal	TRUE/FALSE
StatboxVisible	Statuszeile sichtbar	TRUE/FALSE)
ShowTooltips	Tooltips anzeigen	TRUE/FALSE)
CopyDefType	Was soll kopiert werden	COPYALL COPYFIELD
CopyAsk	Soll nach dem Markieren eine Abfrage erfolgen	TRUE/FALSE
CopyRightBut	Belegung der rechten Maustaste	(MARKFIELD MARKOBJ MARKFRAME SENDKEY)
CopyUseTimeout	Einschalten des Markierens über Timeout	TRUE/FALSE
PrintLeftMargin	Linker Rand beim Drucken	Integer
PrintRightMargin	Rechter Rand beim Drucken	Integer
PrintTopMargin	Oberer Rand beim Drucken	Integer
PrintBottomMargin	Unterer Rand beim Drucken	Integer
PrintAnzPics	Anzahl Bilder pro Seite beim Drucken	Integer

vordefiniert:

Diese Ressourcen sollten nicht in der Ressourcdatei gesetzt werden. Es besteht jedoch die Möglichkeit mittels \$-Ersetzung bei einer Zuweisung auf der rechten Seite oder in einer 'include' Anweisung darauf zuzugreifen.

Ressource	Beschreibung	Werte
GrpDir	Gruppenverzeichnis aus Programmtabelle	String
UsrDir	Benutzerverzeichnis laut Parameter /userdir	String

sonstige

Diese Ressourcen können nur durch das Bearbeiten der Ressourcdatei verändert werden. Sie können sowohl in die Datei mit anwendungsunabhängigen als auch in die Datei mit anwendungsabhängigen Ressourcen geschrieben werden. Bei Werten, die in beide Dateien geschrieben werden, muss die Reihenfolge des Einlesens beachtet werden.

Ressource	Beschreibung	Werte
BeepType	Art der Ausgabe beim Aufruf von <code>tty_beep()</code>	0
BmPackOptions	Kommandozeilenoptionen für das Programm <code>bm-pack</code> , beim Aufruf über den Menüpunkt "Entwickler/Bitmaps packen"	""
BusyTimeout	Zeitverzögerung in Millisekunden bis zum Aufblenden des Busycursors	50
ConnectLogo	Name einer Datei im <code>.bmp</code> Format, die beim Verbindungsaufbau in einem eigenen Fenster angezeigt wird.	STARTUP.BMP
ConnectWave	Name einer Datei im <code>.wav</code> Format, welche beim Verbindungsaufbau über die Soundkarte abgespielt wird	""
CopyStartTimeout	Timeout in Millisekunden zum Einschalten der Bereichsmarkierung	500

DisconnectSignal	Signal, das beim Verbindungsabbau an die Anwendung gesendet wird oder -1 für kein Signal	-1
FirstPort, LastPort	Einschränkung der benutzten TCP/IP-Ports	-1
FixEditor	Programm zum Editieren von Textdateien	notepad.exe
FixEditorCP	Codepage von FixEditor	ANSI
FwBmpTool	Programm zum Editieren von Bitmaps	mspaint.exe
FwEditTool	Programm zum Editieren der Konfigurationsdateien	notepad.exe
FwPrgTool	Programm zum Editieren der Programmtabelle	bin\edprg.exe
ShowJumpCursor	Cursor bei maussensitiven Masken	0
StderrTimeout	Timeout für das Lesen über Stderr.	30
SizeMen_M	Menü für Größe Medium	Größe &1
SizeMen_L	Menü für Größe Large	Größe &2
SizeMen_H	Menü für Größe Huge	Größe &3
StyleMen_0	Menü für Stil 0	Größe &0
StyleMen_1	Menü für Stil 1	Größe &1
StyleMen_2	Menü für Stil 2	Größe &2
StyleMen_3	Menü für Stil 3	Größe &3
TooltipOffsetXField	X-Offset für Tooltip-Positionierung bei Feldern	0
TooltipOffsetYField	Y-Offset für Tooltip-Positionierung bei Feldern	0
TooltipOffsetXPaintarea	X-Offset für Tooltip-Positionierung bei Paintareas	15
TooltipOffsetYPaintarea	Y-Offset für Tooltip-Positionierung bei Paintareas	15
TooltipPositionMeth-Field	Tooltip-Positionierung bei Feldern	FIELD
TooltipPositionMeth-Paintarea	Tooltip-Positionierung bei Paintareas	CURSOR
TooltipTimeout	Zeit in ms bis zum Aufblenden eines Tooltips	3500

Diese Tabelle gibt nur eine Übersicht. Einige Einstellungen erfordern eine genauere Beschreibung, die in den folgenden Abschnitten enthalten ist.

13.4 Ausgabe von Tönen

Das Verhalten von FIX/Win beim Aufruf der FIX-Funktion `tty_beep()` kann über die Ressource

`BeepType`

bestimmt werden. Wenn die Ressource einen Wert > 0 enthält, dann wird dieser Wert als Argument für die Windows-Funktion `MessageBeep()` verwendet. Das bedeutet, dass der diesem Wert zugeordnete Windows-Klang abgespielt wird. Dabei kommen folgende Werte in Betracht:

Wert	Klang
0	Standardton
16	Klang für Kritischer Abbruch
32	Klang für Frage
48	Klang für Hinweis
64	Klang für Stern

Ist der in `BeepType` hinterlegte Wert < 0 , dann enthält er die Frequenz und die Dauer für einen Ton, der über den Lautsprecher ausgegeben wird. Für die Berechnung des Wertes kann folgende Formel verwendet werden:

$(65535 * \text{"Dauer in ms"} + \text{"Frequenz in hz"}) * - 1$

Der Nachteil der Ausgabe über den Lautsprecher liegt darin, dass das Protokoll um die angegebene Dauer blockiert wird, weil der Ton von Windows nicht asynchron abgespielt wird. Der Vorteil liegt darin, dass auch bei der Verwendung von Terminal-Servern ein Ton auf dem Client abgespielt wird.

13.5 Steuerung des Busy-Cursors

Der Busy-Cursor ist ein Mauszeiger, der anzeigt, dass die Anwendung zur Zeit beschäftigt ist und keine Eingaben entgegennehmen kann. Damit dieser Mauszeiger nicht nach jedem Tastendruck angezeigt wird, wird das Einschalten über einen Timeout gesteuert. Der Wert für den Timeout ist in der Ressource

BusyTimeout

zu hinterlegen. Wenn nach der dort angegebenen Zahl in Millisekunden keine weitere Taste von FIX angefordert wird, dann wird der Busy-Cursor eingeblendet.

13.6 Vorgabe von Ports

FIX/Win benötigt zum Verbindungsaufbau zwei Sockets. Der Bereich der Ports, die auf der lokalen (FIX/Win) Seite verwendet werden, kann durch die anwendungsabhängigen Ressourcen

FirstPort

LastPort

eingestellt werden. Wird für einen dieser Werte -1 angegeben, werden die Werte weggelassen oder werden ungültige Werte angegeben, so werden die Ports wie bisher vom Betriebssystem vergeben.

Sind alle Ports in diesem Bereich belegt, wird eine Fehlermeldung ausgegeben (ERR_NT160).

13.7 Definition eines Signals für den Verbindungsabbruch

Damit alle Prozesse beim Beenden einer Verbindung auf Seite der FIX-Anwendung beendet werden, kann über die Ressource

DisconnectSignal

ein Signal eingestellt werden (int Wert), das über den rexec Daemon vor dem Verbindungsabbau an die gestarteten Prozesse gesendet wird. Wird dieser Wert nicht als Ressource definiert oder als -1 festgelegt, dann wird kein Signal gesendet.

13.8 Anzeige eines Logos und Abspielen eines Tons

Durch die Definition der Ressource

ConnectLogo

kann ein Logo im * .bmp-Format definiert werden, das während des Verbindungsaufbaus in einem eigenen Fenster angezeigt wird. Wenn auf diese Weise ein Logo definiert wird, dann kann zusätzlich durch Angabe der Ressource

ConnectWave

eine Datei im * .wav Format angegeben werden, die über die Soundkarte abgespielt wird.

13.9 Timeout bei der Fehlerausgabe

Die Ausgabe des Fehlerkanals der FIX-Anwendung wird von FIX/Win zusammen mit der Meldung FX100 im Meldungsfenster ausgegeben. Vor der Version 3.0.0 wurde von FIX/Win für jeden Leseaufruf eine Fehlermeldung erzeugt. Je nach Konfiguration des Netzwerks kommt es vor, dass eine Ausgabe auf den Fehlerkanal in einem Stück oder in mehreren Teilstücken übertragen wird. Das hatte zur Folge, dass eine Fehlermeldung über mehrere FX100 Meldungen verteilt war. Im schlimmsten Fall wurde für jeden Buchstaben eine FX100 Meldung erzeugt. Um dem entgegen zu wirken kann ab der Version 3.0.0 die Ressource

`StderrTimeout`

gesetzt werden. Sie definiert die Zeit in Millisekunden, die nach dem Lesen von Daten auf dem Fehlerkanal auf weitere Daten gewartet wird. Treffen innerhalb dieses Zeitraums weitere Daten ein, dann werden diese hinter die bereits gelesenen Daten in einen Puffer geschrieben. Danach wird der Timeout erneut gestartet. Erst wenn keine weiteren Daten mehr kommen, wird der Puffer zusammen mit einer FX100 Meldung ausgegeben. Wird bei diesem Verfahren ein zu hoher Wert gewählt, dann besteht die Gefahr, dass mehrere Meldungen zusammen mit einer FX100 Meldung ausgegeben werden. Dies ist jedoch wesentlich besser als die Erzeugung einer Meldung und somit einer Zeile für jeden Buchstaben. Außerdem enthalten die Meldungen, die über den Fehlerkanal ausgegeben werden, Zeilenumbrüche, die auch im Meldungsfenster wiedergegeben werden. Trotzdem besteht die Möglichkeit den mit Version 3.0.0 eingeführten Mechanismus abzuschalten, indem für `StderrTimeout` ein Wert < 0 angegeben wird.

13.10 Zeichensatz von `stderr` und `stdout`

Bei der Verwendung von FIX/Win als Frontend werden die Ausgaben auf den Fehlerkanal der Anwendung (`stderr`) in das Meldungsfenster umgeleitet. Dabei findet eine Umwandlung in den Unicode-Zeichensatz statt. Als Codepage für die von der Anwendung kommenden Zeichen wird dabei die ANSI Codepage angenommen, wenn keine andere Codepage definiert wird. Die Codepage kann durch Setzen der anwendungsabhängigen Ressource

`StderrCP`

explizit definiert werden. Hier ist die Nummer einer auf dem Windows-System installierten Codepage einzutragen. Zusätzlich kann einer der Werte

ANSI

UTF-8

für diese Ressource verwendet werden. Bei der Umstellung einer FIX-Anwendung auf UTF-8 ist es sinnvoll,

`StderrCP: UTF-8`

zu setzen, weil die Meldungen der Anwendung in UTF-8 ausgegeben werden.

Die Texte, die die Anwendung auf den Standardkanal (`stdout`) ausgibt, werden von FIX/Win in eine Datei geschrieben. Dabei werden keinerlei Konvertierungen vorgenommen. Zum Lesen der Datei wird ein externes Programm gestartet. Da dieses Programm in FIX/Win einstellbar ist, ist der Administrator von FIX/Win dafür verantwortlich, ein Programm zu wählen, das den Zeichensatz anzeigen kann, der von der Anwendung auf `stdout` geschrieben wird.

13.11 Konfiguration des Größen-/Stilmenüs

Über die Ressourcen

`StyleMen...`

`SizeMen...`

kann das Menü zur Auswahl von Bildstil und Bildgröße konfiguriert werden. Zu diesen Ressourcen können Texte angegeben werden, die als Menüpunkte verwendet werden. Beginnt ein Text für einen Menüpunkt mit einem "-" Zeichen, dann fehlt dieser Menüpunkt. Beginnt er mit dem Zeichen "*", dann wird der Menüpunkt grau dargestellt.

13.12 Konfiguration von Tooltips

Bestimmung des Timeouts

Die Zeit, nach der ein Tooltip eingeblendet werden soll, ist in der Ressource

`TooltipTimeout`

zu definieren. Die Angabe erfolgt in Millisekunden und der Default bei Nichtangabe des Wertes ist 3500.

Bestimmung der Positionierung

Die Positionierung eines Tooltips kann für Paintareas und Felder getrennt über Ressourcen eingestellt werden. Alle Ressourcen mit `Paintarea` als Namensbestandteil betreffen Paintareas und alle Ressourcen mit `Field` als Namensbestandteil betreffen Felder. Die Positionierung wird durch eine Methode definiert, die in den Ressourcen

`TooltipPositionMethPaintarea`

`TooltipPositionMethField`

einzutragen ist. Wird dort der Wert `CURSOR` eingetragen, dann wird der Tooltip am Zielpunkt (Hotspot) des Mauszeigers zum Zeitpunkt des Aufblendens ausgerichtet. Jeder andere Wert bewirkt, dass der Tooltip an der linken oberen Ecke des Elements (Paintarea oder Feld) ausgerichtet wird. Als Wert für `TooltipPositionMethPaintarea` sollte in diesem Fall `PAINTAREA` verwendet werden und als Wert für `TooltipPositionMethField` der Wert `FIELD`. Der Default für `TooltipPositionMethPaintarea` ist `CURSOR` und der Default für `TooltipPositionMethField` ist `FIELD`.

Zusätzlich zur Methode können in den Ressourcen

`TooltipOffsetXField`

`TooltipOffsetYField`

`TooltipOffsetXPaintarea`

`TooltipOffsetYPaintarea`

Werte angegeben werden, um die der Tooltip in X- und Y-Richtung verschoben wird. Als Default für ein Feld wird 0,0 definiert, so dass der Tooltip direkt über dem Feld plziert wird. Als Default für eine Paintarea wird 15,15 verwendet. Der Tooltip erscheint somit rechts unterhalb des Mauszeigers.

13.13 Starten eines Editors

Dateien können durch den Aufruf der FIX-Funktion `editor()` mit einem Editor bearbeitet werden. Ab der Version FIX 3.1.0 kann diese Funktion auch zusammen mit FIX/Win genutzt werden. Dazu werden die Dateien auf die FIX/Win-Seite kopiert und danach wird ein Editor gestartet. Dabei finden mehrere Konvertierungen statt.

Auf der FIX-Seite wird die Datei in dem Zeichensatz gelesen und geschrieben, der über `FXCHARSET` eingestellt ist. Als Zeilenende wird die unter dem Betriebssystem typische Markierung (`\r\n` unter Windows und `\n` unter UNIX) beim Schreiben verwendet. Beim Lesen sind unter Windows beide Markierungen erlaubt. Unter UNIX muss `\n` an den Zeilenenden stehen.

Nachdem die Daten an FIX/Win gesendet wurden, werden sie in den UNICODE-Zeichensatz konvertiert und in eine temporäre Datei geschrieben. Das Verzeichnis der Datei wird über die Umgebungsvariable `TMP` definiert.

Zum Schreiben wird einer der Zeichensätze:

- ANSI
- UNICODE
- UTF-8

verwendet. Windows konvertiert dabei alle Unicode-Zeichen in diesen Zeichensatz. Der Zeichensatz ist in die anwendungsunabhängige Ressource (`fw_usr.frc`)

`FixEditorCP`

einzutragen. Fehlt die Angabe, dann wird ANSI verwendet. Danach wird ein Editor gestartet, der in der Lage sein muss, diesen Zeichensatz zu lesen und zu schreiben. Die Datei beinhaltet zur Erkennung des Formats die unter Win-

dows übliche Byteordermark am Anfang. Der Pfad des Editors ist in die anwendungsunabhängige Ressource (fw_usr.frc)

FixEditor

einzutragen.

Während der Editor die Datei bearbeitet, wird das FIX/Win Fenster durch einen Dialog gesperrt, der die Namen der Datei (auf FIX- und auf FIX/Win-Seite) anzeigt. Durch das Anklicken der Schaltfläche "Anzeigen" kann der Editor in den Vordergrund gebracht werden. Mit Hilfe der Schaltfläche "Abbrechen" kann der Editor abgebrochen werden.

Wenn der Editor beendet wird und die Datei verändert wurde, dann wird die Datei wieder in FIX/Win eingelesen. Die Datei muss dabei den in FixEditorCP vereinbarten Zeichensatz besitzen. Nach dem Einlesen werden die Zeichen zu FIX gesendet und in den durch FXCHARSET definierten Zeichensatz umgewandelt. FIX schreibt die Daten dann in die angegebene Datei.

Wurde die Datei nicht verändert, dann wird FIX lediglich über das Ende des Editiervorgangs informiert und der Aufruf von editor() kehrt zurück.

In beiden Fällen wird der Dialog über dem FIX/Win Fenster entfernt.

4 Erstellen einer Benutzerbibliothek

1 Allgemeine Hinweise

1.1 Laden der Benutzerbibliothek

Um eine Benutzerbibliothek beim Start des Standard-Frameworks von FIX/Win zu laden, ist die Bibliothek zusammen mit dem Schalter

```
/fwown <Dateipfad>
```

anzugeben. Um festzustellen, welche Bibliothek FIX/Win benutzt, kann der Menüpunkt *Hilfe/Info* aufgerufen werden. Er zeigt außer einer Copyright-Meldung den Namen der verwendeten Bibliothek und eine Versionsinformation an.

1.2 Include-Dateien

Alle Datentypen, Definitionen, Makros und Prototypen zur Erstellung einer Benutzerbibliothek sind in den beiden Dateien

```
include/fwown.h  
include/fwownStd.h
```

definiert. Die Datei `include/fwown.h` enthält alle Definitionen, die sich auf die Kernklassen (`fixwinCore.dll`) beziehen. In der Datei `include/fwownStd.h` stehen zusätzliche Definitionen, die sich auf Funktionen des Standard-Frameworks (`fixwin.exe`) beziehen. Wenn ein eigenes Framework erstellt wird und auf Funktionen dieses Frameworks zugegriffen werden soll, dann ist eine weitere Datei mit den entsprechenden Definitionen bereitzustellen.

1.3 Arten von Funktionen

Bei der Verwendung einer Benutzerbibliothek sind zwei Arten von Funktionen relevant:

- Funktionen, deren Code in der Benutzerbibliothek liegt. Diese Funktionen werden im kommenden Text *FWown-Funktionen* genannt. Sie werden von FIX/Win zu bestimmten Zeitpunkten aufgerufen. Damit dies funktioniert, muss die Benutzerbibliothek diese Funktionen exportieren.
- Funktionen, deren Code in FIX/Win liegt (entweder in den Kernklassen oder im Framework) und die der Benutzerbibliothek zur Verfügung gestellt werden. Diese Funktionen werden im folgenden Text *FIX/Win-Funktionen* genannt. Sie können von der Benutzerbibliothek aufgerufen werden.

Da die Definition der FWown-Funktionen sowohl FIX/Win als auch der Benutzerbibliothek bekannt sein müssen, werden die Dateien `fwown.h` und `fwownStd.h` von beiden verwendet. Bei der Verwendung in dem Code der Benutzerbibliothek ist es wichtig, dass vor dem Einschließen dieser Dateien das Makro

```
#define FWOWN_EXPORTS
```

gesetzt wird, damit die Funktionen richtig definiert werden.

1.4 Parameter Callback

Alle FWown-Funktionen bekommen den Parameter `Callback`. Er ermöglicht den Zugriff auf FIX/Win-Funktionen. Dazu enthält er einen Zeiger auf eine Funktion innerhalb des FIX/Win Codes. Durch Aufruf dieser Funktion mit einer bestimmten Konstante als Parameter können Zeiger auf weitere Funktionen von FIX/Win ermittelt werden. Die Konstanten, die als Parameter verwendet werden können, sind in den Dateien `fown.h` und `fownStd.h` als Makros definiert. Der Rückgabewert der Callback-Funktion ist in einen Funktionszeiger des entsprechenden Typs zu casten.

Um die Programmierung zu erleichtern wird in `fown.h` für jede Funktion ein Makro definiert, das als Parameter den Callback und die Variable, die den Funktionspointer aufnehmen soll, bekommt. Das Makro prüft, ob die Variable bereits mit einem Funktionspointer belegt ist (`!= NULL`) und ruft ansonsten die Callback Funktion zur Ermittlung des Funktionspointers auf. Bei der Zuweisung an die Variable wird dann in den richtigen Typ gecastet. In den folgenden Beschreibungen der FIX/Win-Funktionen werden neben dem Prototyp das Makro für die Konstante der Callback-Funktion und das Makro zum Aufruf der Callback-Funktion angegeben.

Beispiel:

Der folgende Code zeigt, wie beim Aufruf der Funktion `FwownExec()` der Benutzername ermittelt werden kann:

```
static StringInfoFkt GetUsername = NULL;

FWOWN_API _TCHAR* CALLBACK FwownExec(CallBackFkt Callback, void* p_callID, _TCHAR* str)
{
    _TCHAR* username;

    FWFKT_GETUSER(Callback, GetUsername);

    username = GetUsername(p_callID);
}
```

Der Zeiger auf die FIX/Win-Funktion `GetUsername()` wird in der statischen Variablen `GetUsername` gehalten. Da die Variable zu Beginn auf `NULL` zeigt, ruft das Makro `FWFKT_GETUSER` die Funktion `Callback` zur Ermittlung des Funktionszeigers auf. Bei einem weiteren Aufruf von `FwownExec()` (oder einer anderen FWown-Funktion) erfolgt dieser Schritt nicht mehr, weil der Wert in `GetUsername != NULL` ist. Es ist jedoch auch nicht notwendig, weil sich die Adresse einer FIX/Win-Funktion nicht zwischen zwei Aufrufen ändern kann. Damit diese Optimierung funktioniert, ist es wichtig, dass `GetUsername` als statische globale Variable definiert wird, die mit `NULL` initialisiert wird.

1.5 Parameter p_callID

Der Parameter `p_callID` wurde mit der Version 3.0.0 eingeführt. Er ist, genauso wie der Parameter `Callback`, bei allen FWown-Funktionen vorhanden. Außerdem besitzt jede FIX/Win-Funktion diesen Parameter als ersten Parameter. Beim Aufruf einer FIX/Win-Funktion ist der Wert, den die FWown-Funktion von FIX/Win bekommen hat, an die FIX/Win-Funktion weiterzugeben. Ansonsten muss sich der Entwickler der Benutzerbibliothek nicht um diesen Parameter kümmern.

Der Grund für die Einführung dieses Parameters hängt damit zusammen, dass FIX/Win mit der Version 3.0.0 es erlaubt, mehrere interne Fenster mit jeweils einer FIX/Win Anwendung zu öffnen. Dazu wird für jedes Fenster eine neue Instanz einer Klasse erzeugt. Wenn eine FIX/Win-Funktion aufgerufen wird, muss eine Möglichkeit bestehen, festzustellen, um welche Instanz es sich handelt.

2 Beschreibung der FWown-Funktionen

FwownGetVersionInfo - Versionsinfo ermitteln

Definition

```
extern FWOWN_API _TCHAR* CALLBACK FwownGetVersionInfo(CallBackFkt p_callBack,
                                                    void* p_callID, int* version);
```

Aufrufer

Standard-Framework, Kernklassen

Aufrufzeitpunkt

Beim Aufblenden des Dialogs und beim Verbindungsaufbau.

Beschreibung

Die Funktion wird zur Ermittlung einer Versionsangabe aufgerufen, die in dem Dialog angezeigt wird, der durch den Menüpunkt "Hilfe/Info" im Standard-Framework gestartet wird. Die Funktion liefert als Ergebnis einen (selbst zu definierenden) Versionsstring und einen (selbst zu definierenden) Featurelevel, der in der Variablen *version* abgelegt werden muss. Auf der FIX Seite können diese Werte mittels

```
_TCHAR *fx_fwown_get_version_info(int *feature_level);
```

abgefragt und für eigene Auswertungen genutzt werden. Dazu wird die Funktion unmittelbar nach dem Verbindungsaufbau aufgerufen. Besitzt die Benutzerbibliothek nicht die Funktion *FwownGetVersionInfo*, dann wird eine leere Zeichenkette("") als Rückgabewert geliefert und 0 nach *feature_level* geschrieben.

FwownExec - Backend Request

Definition

```
FWOWN_API _TCHAR* CALLBACK FwownExec(CallBackFkt p_callBack, void* p_callID, _TCHAR* str);
```

Aufrufer

Kernklassen

Aufrufzeitpunkt

Wenn die Funktion *fx_exec_frontend(_TCHAR* str)* von der FIX-Anwendung aufgerufen wird.

Beschreibung

Die von der FIX-Anwendung übergebene Zeichenkette wird an die Funktion *FwownExec()* als Parameter *str* übergeben. Diese kann den String analysieren und aufgrund des Inhaltes bestimmte Aktionen ausführen. Als Rückgabewert muss sie entweder NULL im Erfolgsfall oder einen Zeiger auf eine Zeichenkette mit einer Fehlermeldung liefern. Die Fehlermeldung wird von FIX/Win im Meldungsfenster ausgegeben. Die Größe der Zeichenkette in *str* ist durch den Protokollpuffer von FIX/Win auf ca. 8000 Bytes¹ begrenzt und es dürfen nur druckbare Zeichen verwendet werden.

FwownProcessData - Binäre Daten verarbeiten

Definition

```
FWOWN_API _TCHAR* CALLBACK FwownProcessData(CallBackFkt p_callBack, void* p_callID,
                                             long* p_bytes, char** p_data);
```

Aufrufer

Kernklassen

1. Eine genaue Angabe ist nicht möglich, da der Puffer je nach Situation auch noch andere Daten enthält.

Aufrufzeitpunkt

Wenn bei dem Anfordern einer Taste Daten von FIX übertragen wurden.

Beschreibung

Im Gegensatz zu `FwownExec()` können von dieser Funktion binäre Daten beliebiger Länge empfangen werden. Weiterhin ist es möglich, die Daten nach einer Veränderung zurück an die FIX-Anwendung zu übermitteln. Aufgerufen wird diese Funktion als Reaktion auf die FIX Funktion `fx_transfer_to_frontend(int bytes, char* data)`. Da die Daten stückchenweise über den Protokollpuffer gesendet werden müssen, wird die Funktion erst aufgerufen, wenn FIX eine Taste anfordert. Denn nur in diesem Zustand ist es möglich, die veränderten Daten als Antwort an das FIX Programm zurückzusenden.

Der Parameter `bytes` enthält einen Zeiger auf die Länge der Daten. Dieser Wert kann von der Funktion geändert werden. Ist er nach Ablauf der Funktion > 0 , dann wird diese Menge an Daten an das FIX Programm zurückgesendet.

Der Parameter `data` enthält einen Zeiger auf die Adresse der Daten. Auch dieser Wert kann geändert werden, was sinnvoll ist, wenn ein größerer Datenbereich benötigt wird. Die Adresse muss jedoch im Speicherbereich von FIX/Win liegen. Deshalb ist ein neuer Puffer mit der Funktion `ResizeBuffer()` (kann durch Aufruf von `FWFKT_RESIZEBUFFER` ermittelt werden) zu allozieren. Sie bekommt als Parameter den alten Speicherbereich und die neue Größe und liefert die Adresse des neuen Puffers.

Da das FIX/Programm zum Abarbeitungszeitpunkt der Funktion `FwownProcessData()` sich in der Tastaturabfrage befindet, kann der veränderte Puffer nicht als Rückgabewert der Funktion `fx_transfer_to_frontend()` zurückgeliefert werden. Statt dessen bekommt die FIX Anwendung das Event `K_SB`. Sie kann bei der Abarbeitung dieses Events mittels `getFrontendData(char** data)` den Zeiger auf die Daten (Parameter `data`) und die Länge (Rückgabewert) ermitteln. Diese Daten stehen jedoch nach der Abarbeitung des Events nicht mehr zur Verfügung und sollten deshalb, wenn notwendig, in einen eigenen Puffer kopiert werden.

FwownDrawPaintArea - PaintArea zeichnen

Definition

```
FWOWN_API _TCHAR* CALLBACK FwownDrawPaintArea(CallBackFkt p_callBack, void* p_callID,
                                             HDC p_hDCPA, int p_width, int p_height,
                                             unsigned long p_pa_id, short p_pa_type,
                                             _TCHAR* p_text, unsigned _TCHAR p_pa_attr, short p_pa_align,
                                             long p_longval1, long p_longval2, long p_longval3, long p_longval4,
                                             _TCHAR* p_ms_name, short p_objclass,
                                             int p_size, int p_style)
```

Aufrufer

Kernklassen

Aufrufzeitpunkt

Wenn eine Paintarea oder ein Teil einer Paintarea sich geändert hat und neu gezeichnet werden muss.

Beschreibung

Die Funktion hat die Aufgabe eine Paintarea zu zeichnen. Die Parameter haben folgende Bedeutung:

- *p_callBack und p_callID*
Dies sind die üblichen Parameter zum Aufruf von FIX/Win-Funktionen, wie sie an alle Funktionen der Benutzerbibliothek übergeben werden.
- *p_hDCPA*
Dieser Wert ist das Handle der Zeichenfläche, die gefüllt werden muss. In `p_width` und `p_height` wird die Größe der Fläche angegeben. Die Funktion `FwownDrawPaintArea()` muss immer die gesamte Zeichenfläche füllen. FIX/Win ist dafür verantwortlich, nur den benötigten Teil in den Bildschirm zu kopieren, so dass eine darüber liegende Maske nicht davon betroffen wird.
- *p_pa_id*
`p_pa_id` ist eine eindeutige Nummer für die Paintarea. Mit Hilfe dieser Nummer ist es möglich, Daten zu einer Paintarea in der Benutzerbibliothek zu verwalten und zum Darstellen zu nutzen.

- *p_pa_type*
Der Wert in `p_pa_type` beinhaltet den Typ der Paintarea. Es kann eine der folgenden Konstanten aus `frown.h` sein:
 - `PA_TEXTLABEL` - ein Textlabel.
 - `PA_TABLEHEADER` - ein Text als Kopf einer Tabellenspalte.
 - `PA_BITMAP` - eine Bitmap.
 - `PA_BUTTON` - ein Button.
 - `PA_VARBUTTON` - ein Button zur Umschaltung von Varianten.
 - `PA_USERDEFINED` - ein benutzerdefinierter Typ.
 Je nach Typ der Paintarea sollte `FwownDrawPaintArea()` eine andere Zeichenroutine zur Darstellung wählen.
- *p_text, p_pa_attr*
Der Parameter `p_text` enthält den übersetzten Text der Paintarea. In `p_pa_attr` steht das Textattribut dazu. Abgesehen von dem Typ `PA_BUTTON` enthält diese Variable den durch `pa_declare()` oder `pa_put()` definierten Wert.
- *p_align*
Die Ausrichtung des Textes wird durch den Wert in `p_align` bestimmt. Folgende Werte sind möglich:
 - `PA_ALGN_LEFT` - Der Text soll linksbündig ausgerichtet werden.
 - `PA_ALGN_CENTER` - Der Text soll zentriert werden.
 - `PA_ALGN_RIGHT` - Der Text soll rechtsbündig ausgerichtet werden.
- *p_longval1, p_longval2, p_longval3 und p_longval4*
Die Werte in `p_longval1`, `p_longval2`, `p_longval3` und `p_longval4` stammen aus den gleichnamigen Parametern, die bei `pa_put()` oder `pa_declare()` anzugeben sind. Zusammen mit `p_ms_name` und `p_objclass`, die den Namen und die Objektklasse des Objekts beinhalten, dem die Paintarea zugeordnet ist, können diese Werte verwendet werden, um davon gewisse Abweichungen beim Zeichnen abzuleiten. Einige der Werte werden von FIX bereits mit einer festen Bedeutung versehen (siehe unten bei den Besonderheiten der verschiedenen Typen).
- *p_size, p_style*
Um Paintareas je nach Look&Feel anders darstellen zu können, wird in `p_size` die aktuelle Größe (0-2) und in `p_style` der aktuelle Stil (0-3) übergeben.

Beim Aufruf der Funktion `FwownDrawPaintArea()` kann davon ausgegangen werden, dass vorher:

- Die zum Attribut passende Zeichenfarbe eingestellt wurde.
- Die zum Attribut passende Hintergrundfarbe eingestellt wurde.
- Der Hintergrund durch Zeichnen eines Rechtecks in dieser Farbe gelöscht wurde.
- Der Ausgabemodus auf `TRANSPARENT` geschaltet wurde (Windows-Funktion `SetBkMode()`), damit der Hintergrund durchscheint.
- Ein Font der Schriftart "MS Sans Serif" in der passenden Schriftgröße eingestellt wurde.

Die Farbe für Vordergrund und Hintergrund wird aus der Farbzusatztabelle gelesen. Je nach Typ der Paintarea und je nach Attribut wird ein anderer Eintrag verwendet. Der Name des Eintrags wird im Abschnitt [Erstellen eigener Farbzusatztabellen](#) auf Seite 63 beschrieben.

Für den Rückgabewert der Funktion gibt es drei verschiedene Möglichkeiten:

- Es ist ein Fehler aufgetreten: Der Rückgabewert muss auf eine Fehlermeldung zeigen, die von FIX/Win ausgegeben wird.
- Der Bereich konnte von der Funktion gefüllt werden: Als Rückgabewert muss `NULL` verwendet werden.
- Die Paintarea wurde nicht von der Funktion der Benutzerbibliothek gefüllt: Als Rückgabewert muss "" (leere Zeichenkette) verwendet werden. FIX/Win übernimmt in diesem Fall die Darstellung durch eine Standardmethode.

Besonderheiten des Typs PA_BUTTON

Bei dem Typ PA_BUTTON bestimmt FIX/Win das Attribut in `p_pa_attr`, wenn es nicht von `pa_declare()` oder `pa_put()` als GRAYED definiert wurde. Dazu wird der Zustand des Buttons ausgewertet. Die möglichen Zustände und die zugeordneten Attribute sind wie folgt definiert:

- "Normal" - wie von `pa_declare()` oder `pa_put()` definiert.
- "Mauszeiger über Paintarea" - A_BOLD
- "Button gedrückt" - A_INVERS

Besonderheiten des Typs PA_TABLEHEADER

Für den Typ PA_TABLEHEADER definiert FIX die folgenden Werte für `p_longval1`:

- PA_TABLEHEADER_FIRST markiert den Kopf der ersten und der folgenden Tabellenspalten.
- PA_TABLEHEADER_LAST markiert den Kopf der letzten Tabellenspalte, der anders gezeichnet werden muss als die der ersten Spalten.

Besonderheiten des Typs PA_VARBUTTON

Für den Typ PA_VARBUTTON enthält `longval1` in den Bits 8-9 die Ausrichtung und in den übrigen Bits die Art des Buttons. Für die Art kommen folgende Werte in Betracht:

- OL - normaler Varbutton
- PA_VBTN_ACTIVE - aktiver Varbutton
- PA_VBTN_SCRL_FWRD - Varbutton, um nach vorne zu scrollen.
- PA_VBTN_SCRL_BWRD - Varbutton, um nach hinten zu scrollen.

Hierbei ist zu beachten, dass in den Bits 8-9 von `longval1` die Ausrichtung kodiert ist. Daher ist vor einem Vergleich der Wert von `longval1` mit dem Komplementärwert von

- PA_VBTN_PLMT_BITS

zu verknüpfen (`longval1 & ~PA_VBTN_PLMT_BITS`). Als Wert für die Ausrichtung kommen folgende Werte in Betracht:

- PA_VBTN_PLMT_TOP - oben liegender Reiter.
- PA_VBTN_PLMT_BOTTOM - unten liegender Reiter.
- PA_VBTN_PLMT_LEFT - links liegender Reiter.
- PA_VBTN_PLMT_RIGHT - rechts liegender Reiter.

Um hier nur die Bits 8-9 für einen Vergleich zu erhalten ist mit dem Wert PA_VBTN_PLMT_BITS zu verknüpfen (`longval1 & PA_VBTN_PLMT_BITS`). Das Zeichnen von links und rechts liegenden Reitern ist nicht notwendig, da diese von FIX z.Z. noch nicht unterstützt werden.

Der folgende Code zeigt ein Grundgerüst, das den Wert in `longval1` auswertet:

```
long placement;
long subtype;

subtype = longval1 & ~PA_VBTN_PLMT_BITS;
placement = longval1 & PA_VBTN_PLMT_BITS;

switch (placement) {
case PA_VBTN_PLMT_BOTTOM:
    switch (subtype) {
    case PA_VBTN_ACTIVE:
        /* Zeichne aktiven Varbutton mit Ausrichtung unten */
        break;
    case PA_VBTN_SCRL_FWRD:
        /* Zeichne Varbutton zum Vorwärtsscrollen mit Ausrichtung unten */
        break;
    case PA_VBTN_SCRL_BWRD:
        /* Zeichne Varbutton zum Rückwärtsscrollen mit Ausrichtung unten */
        break;
    default:
```

```

        /* Zeichne normalen Varbutton mit Ausrichtung unten */
        break;
    }
    break;
case PA_VBTN_PLMT_TOP:
    switch (subtype) {
    case PA_VBTN_ACTIVE:
        /* Zeichne aktiven Varbutton mit Ausrichtung oben */
        break;
    case PA_VBTN_SCRL_FWD:
        /* Zeichne Varbutton zum Vorwärtsscrollen mit Ausrichtung oben */
        break;
    case PA_VBTN_SCRL_BWRD:
        /* Zeichne Varbutton zum Rückwärtsscrollen mit Ausrichtung oben */
        break;
    default:
        /* Zeichne normalen Varbutton mit Ausrichtung oben */
        break;
    }
    break;
}
}

```

Neben dem Zeichnen der Varbuttons ist es notwendig, den Rahmen so darzustellen, dass er zur Art der Varbuttons passt. Dazu ist mit Semigrafikzeichen ein Rahmen um den Bereich zu zeichnen, der auch die Leiste mit den Varbuttons in der oberen Zeile (unteren Zeile bei PA_VBTN_PLMT_BOTTOM) enthält. Für die Semigrafikzeichen sind typischerweise Codes zu verwenden, die bisher noch nicht in der Anwendung genutzt wurden. Für diese Codes sind Bitmaps in FIX/Win zu erstellen, die zu den Varbuttons in Ausrichtung und Farbe passen. Die Varbuttons werden auf diesen Rahmen platziert. Die Funktionen von FIX sorgen dafür, dass die Zeichen unten den Varbuttons gerettet werden und beim Entfernen des Varbuttons oder beim Scrollen der Leiste wieder dargestellt werden.

FwownAddPaintArea - Paintarea wurde angelegt

Definition

```

FWOWN_API_TCHAR* CALLBACK FwownAddPaintArea(CallBackFkt p_callBack, void* p_callID,
                                             unsigned long p_pa_id, short p_pa_type, _TCHAR* p_text,
                                             int p_pa_len, short p_pa_align,
                                             long p_longval1, long p_longval2, long p_longval3, long p_longval4)

```

Aufrufer

Kernklassen

Aufrufzeitpunkt

Wenn in FIX/Win einen neue Paintarea angelegt wird.

Beschreibung

Die Funktion wird aufgerufen, wenn eine Paintarea neu angelegt wurde. Die Benutzerbibliothek hat somit die Möglichkeit zu einer Paintarea weitere Daten zu verwalten. Als Schlüssel für die Daten kann die eindeutige Nummer der Paintarea in p_pa_id verwendet werden. Eine Beschreibung der anderen Parameter ist bei der Beschreibung der Funktion [FwownDrawPaintArea - PaintArea zeichnen](#) auf Seite 92 zu finden.

FwownUpdatePaintArea - Paintarea wurde aktualisiert

Definition

```

FWOWN_API_TCHAR* CALLBACK FwownUpdatePaintArea(CallBackFkt p_callBack, void* p_callID,
                                                unsigned long p_pa_id, short p_pa_type, _TCHAR* p_text,
                                                short p_pa_align,
                                                long p_longval1, long p_longval2, long p_longval3, long p_longval4)

```

Aufrufer

Kernklassen

Aufrufzeitpunkt

Beim Verändern einer Paintarea.

Beschreibung

Die Funktion wird aufgerufen, wenn eine Paintarea verändert wurde. Die Benutzerbibliothek hat somit die Möglichkeit zu einer Paintarea weitere Daten zu verwalten. Als Schlüssel für die Daten kann die eindeutige Nummer der Paintarea in `p_pa_id` verwendet werden. Eine Beschreibung der anderen Parameter ist bei der Beschreibung der Funktion [FwownDrawPaintArea - PaintArea zeichnen](#) auf Seite 92 zu finden.

FwownDelPaintArea - Paintarea wurde gelöschtDefinition

```
extern FWOWN_API_TCHAR* CALLBACK FwownDelPaintArea(CallBackFkt p_callBack, void* p_callID,
                                                unsigned long p_pa_id);
```

Aufrufer

Kernklassen

Aufrufzeitpunkt

Nach dem Löschen einer Paintarea.

Beschreibung

Die Funktion wird aufgerufen, wenn eine Paintarea gelöscht wurde. Die Benutzerbibliothek hat somit die Möglichkeit zu einer Paintarea weitere Daten zu verwalten. Als Schlüssel für die Daten kann die eindeutige Nummer der Paintarea in `p_pa_id` verwendet werden.

FwownHook - FIX/Win hat bestimmte Codestelle erreichtDefinition

```
FWOWN_API_TCHAR* CALLBACK FwownHook(CallBackFkt p_callBack, void* p_callID, int p_id)
```

Aufrufer

Standard-Framework, Kernklassen

Aufrufzeitpunkt

Bei verschiedenen Ereignissen in FIX/Win.

Beschreibung

Die Funktion wird von FIX/Win aufgerufen, um die Benutzerbibliothek über bestimmte Ereignisse zu informieren. Die Art des Ereignisses wird in Form des Integerwertes `p_id` mitgegeben. Als Rückgabewert kann die Funktionen einen Text zurückgeben (Fehlermeldung), der im Meldungsfenster angezeigt wird. Ansonsten ist NULL zurückzugeben. Die möglichen Werte für `p_id` werden in `frown.h` und `frownStd.h` als Makros definiert. Für folgende Ereignisse erfolgt ein Aufruf von `FwownHook()`:

p_id	Beschreibung
HOOK_CONNECT	Erfolgreicher Verbindungsaufbau.
HOOK_DISCONNECT	Erfolgreicher Verbindungsabbau.
HOOK_CONNECT_CLOSE	Erfolgreicher Verbindungsabbau beim Schließen des Fensters.
HOOK_PROTOINIT	Protokoll initialisiert.
HOOK_CHANGESIZE	Bildgröße wurde geändert.
HOOK_CHANGESTYLE	Bildstil wurde geändert.
HOOK_EXEC_PROGRAM	Weiteres FIX Programm wurde mittels <code>fxExecProgram()</code> gestartet.
HOOK_EXEC_END	Mit <code>fxExecProgram()</code> gestartetes Programm wurde beendet.

p_id	Beschreibung
HOOK_START	FIX/Win gestartet (Aufruf unmittelbar, nachdem die Benutzerbibliothek geladen wurde)
HOOK_END	Unmittelbar vor dem Beenden von FIX/Win.
HOOK_BEFORE_SAVE_INI	Vor dem Speichern der Einstellungen.
HOOK_AFTER_SAVE_INI	Nach dem Speichern der Einstellungen.

Im Rahmen der Bereitstellung von FIX/Win als DLL wurden die Werte ab Version 3.0.0 wie folgt aufgeteilt:

- Hooks die von den FIX/Win Kernklassen ausgelöst werden.
- Hooks die vom Standard-Framework ausgelöst werden.

Die IDs für Hooks die von den FIX/Win Kernklassen ausgelöst werden, sind in der Datei `fwown.h` definiert. Die anderen sind in `fwownStd.h` definiert.

Da das Standard-Framework das FIX/Win-Fenster erst beim Aufbau einer Verbindung erzeugt, kann es vorkommen, dass der Parameter `p_callID` bei folgenden Hooks mit dem Wert NULL belegt wird:

```
HOOK_START
HOOK_END
```

Weiterhin hat das zur Folge, dass die als `Callback` mitgegebene Callback-Funktion nur für die folgenden IDs einen gültigen Funktionszeiger zurückliefert:

```
FW_GETPRGTAB
FW_GETUSERDIR
FW_GETMSGHWND
FW_GETMAINFRAMEHWND
FW_GETCLIENTHWND
FW_GETCONTAINERHWND
FW_MSGWINSHOWMSG
FW_STATBOXSETMSG
```

Weiterhin benötigen die Funktionen, die über die IDs

```
FW_GETCONTAINERHWND
FW_STATBOXSETMSG
```

ermittelt werden, einen gültigen Wert im Parameter `p_callID`, so dass diese Funktionen innerhalb der Abarbeitung der Hooks `HOOK_START` und `HOOK_END` zwar aufgerufen werden können, jedoch kein sinnvolles Ergebnis liefern.

3 Beschreibung der FIX/Win-Funktionen

Eine Funktion von FIX/Win kann entweder im Kern von FIX/Win liegen oder im Framework. Dieses Handbuch beschreibt nur die Funktionen des Kerns und des Standard-Frameworks. Wenn mit dem Kern ein eigenes Framework erstellt wird, dann stehen nur die Funktionen des Kerns zur Verfügung, es sei denn, das eigene Framework bietet selbst Funktionen an, die von der Benutzerbibliothek aufgerufen werden können.

3.1 Funktionen des FIX/Win Kerns

Alle Prototypen und Definitionen sind in `include/fwown.h` definiert.

GetFrameResInst - Instanz der Ressource-DLL des Frameworks ermitteln

Definition

HINSTANCE GetFrameResInst(void* p_callID)

Makros

FW_GETFRAMERESINST, FWFKT_GETFRAMERESINST(callback, varname)

Beschreibung

Die Funktion liefert das Instanz-Handle der DLL mit den Ressourcen des FIX/Win Frameworks.

GetUsername - Benutzername ermitteln

Definition

_TCHAR* GetUsername(void* p_callID)

Makros

FW_GETUSER, FWFKT_GETUSER(callback, varname)

Beschreibung

Die Funktion liefert den Namen des Benutzers, der sich an der FIX-Anwendung angemeldet hat.

GetPassword - Passwort ermitteln

Definition

_TCHAR* GetPassword(void* p_callID)

Makros

FW_GETPASS, FWFKT_GETPASS(callback, varname)

Beschreibung

Die Funktion liefert das Passwort des Benutzers, der sich an der FIX-Anwendung angemeldet hat.

GetHostname - Hostname ermitteln

Definition

_TCHAR* GetHostname(void* p_callID)

Makros

FW_GETHOSTNAME, FWFKT_GETHOSTNAME(callback, varname)

Beschreibung

Die Funktion liefert den Hostnamen des Rechners, an den sich FIX/Win angemeldet hat. Das Format des Hostnamens entspricht dem Format, das zum Verbindungsaufbau verwendet wurde. Für das Standard-Framework bedeutet das, dass der in der Programmtabelle hinterlegte Namen verwendet wird. Wird statt des Namens eine IP-Adresse zum Verbindungsaufbau verwendet, dann liefert die Funktion ebenfalls eine IP-Adresse. Beinhaltet die Angabe den Port, dann liefert die Funktion auch die durch ":" getrennte Portangabe.

GetGroup - Gruppenverzeichnis ermittelnDefinition

_TCHAR* GetGroup(void* p_callID)

Makros

FW_GETGROUP, FWFKT_GETGROUP(callback, varname)

Beschreibung

Die Funktion liefert das Gruppenverzeichnis, das bei der Anmeldung verwendet wurde.

GetFixHwnd - Handle des FIX/Win-Fensters ermittelnDefinition

HWND GetFixHwnd(void* p_callID)

Makros

FW_GETFIXHWND, FWFKT_GETFIXHWND(callback, varname)

Beschreibung

Die Funktion liefert das Handle des FIX/Win-Fensters.

GetIconHwnd - Handle des IconBox-Fensters ermittelnDefinition

HWND GetIconHwnd(void* p_callID)

Makros

FW_GETICONHWND, FWFKT_GETICONHWND(callback, varname)

Beschreibung

Die Funktion liefert das Handle des IconBox-Fensters.

GetResInstance - Instanz der Ressource-DLL ermittelnDefinition

HINSTANCE GetResInstance(void* p_callID)

Makros

FW_GETRESINSTANCE, FWFKT_GETRESINSTANCE(callback, varname)

Beschreibung

Die Funktion liefert das Instanz-Handle der DLL mit den Ressourcen des FIX/Win Kerns.

GetScreenInfo - Informationen zu Bildeinstellung ermitteln

Definition

```
void GetScreenInfo(void* p_callID, int* style, int* size, int* sx, int* sy)
```

Makros

```
FW_GETSCREENINFO, FWFKT_GETSCREENINFO(callback, varname)
```

Beschreibung

Mit dieser Funktion können die Parameter der aktuellen Bildeinstellung ermittelt werden. Sie bekommt als Parameter Zeiger auf Variablen, in denen die Werte abgelegt werden. `style` ist der aktuelle Stil des Bildschirms. `size` ist die Nummer der Größe (0=medium, 1=large, 2=huge). In `sx` und `sy` werden die Ausmaße einer Zelle dieser Größe in Pixeln abgelegt.

GetColor - Farbinformation aus Farbzordnungstabelle lesen

Definition

```
COLORREF GetColor(void* p_callID, ColtabIdx idx, bool foreground)
```

Makros

```
FW_GETCOLOR, FWFKT_GETCOLOR(callback, varname)
```

Beschreibung

Mit Hilfe dieser Funktion kann auf die Farbzordnungstabelle zugegriffen werden, die aus der Datei `coltab.fix` gelesen wird. Der Parameter `idx` bestimmt das Element, dessen Farbe ermittelt werden soll. Die möglichen Werte werden in `enum ColtabIdx` definiert (`fwown.h`). Der Parameter `foreground` bestimmt, ob die Vordergrundfarbe (`TRUE`) oder die Hintergrundfarbe (`FALSE`) ermittelt werden soll. Als Ergebnis liefert die Funktion die Farbe als `COLORREF` zurück. Dieser Wert kann sowohl zur Erzeugung einer Pinsel- oder Linienfarbe verwendet werden, als auch zur Definition der Hinter-/Vordergrundfarbe eines Textes.

GetCharset- Zeichensatzinformation lesen

Definition

```
int GetCharset(void* p_callID)
```

Makros

```
FW_GETCHARSET, FWFKT_GETCHARSET(callback, varname)
```

Beschreibung

Mit dieser Funktion kann der aktuelle Zeichensatz ermittelt werden. Als Ergebnis liefert die Funktion einen der Werte `CHARSET_LATIN_1` oder `CHARSET_LATIN_2`. Da der Zeichensatz während des Verbindungsaufbaus umgeschaltet werden kann, sollte ein neuer Zeichensatz bei Ausführung von `FwownHook()` mit der ID `HOOK_PROTOINIT` neu ermittelt werden.

SendEvent- Event an FIX-Anwendung senden

Definition

```
void SendEvent(void* p_callID, int ev)
```

Makros

FW_SENDEVENT, FWFKT_SENDEVENT(callback, varname)

Beschreibung

Die Funktion kann zum Senden eines Events an die FIX-Anwendung aufgerufen werden. Das Event ist als Parameter *ev* zu übergeben.

ResizeBuffer - Puffergröße ändernDefinition

_TCHAR* ResizeBuffer(void* p_callID, _TCHAR* buffer, long size)

Makros

FW_RESIZEBUFFER, FWFKT_RESIZEBUFFER(callback, varname)

Beschreibung

Die Funktion `ResizeBuffer()` kann verwendet werden, um einen Speicherbereich, der von FIX/Win übergeben wurde, zu vergrößern. Die Funktion bekommt den Zeiger auf den Speicherbereich (`buffer`) und die gewünschte Größe (`size`). Sie liefert einen Zeiger auf den neuen Speicherbereich zurück. Die Funktion wird typischerweise innerhalb der Funktion `FwownProcessData()` aufgerufen.

3.2 Funktionen des FIX/Win Standard-Frameworks

Alle Prototypen und Definitionen sind in `include/fwownStd.h` definiert.

GetPrgtab - Pfad der Programmtabelle ermittelnDefinition

_TCHAR* GetPrgtab(void* p_callID)

Makros

FW_GETPRGTAB, FWFKT_GETPRGTAB(callback, varname)

Beschreibung

Die Funktion liefert den Pfad der Datei mit der Programmtabelle.

GetUserDir - Pfad des Benutzerverzeichnisses ermittelnDefinition

_TCHAR* GetUserDir(void* p_callID)

Makros

FW_GETUSERDIR, FWFKT_GETUSERDIR(callback, varname)

Beschreibung

Die Funktion liefert den Pfad des Benutzerverzeichnisses. (siehe Kommandozeilenparameter */userdir <Path>* [auf Seite 21](#))

GetMsgWinHwnd - Handle des Meldungsfensters ermitteln

Definition

HWND GetMsgWinHwnd(void* p_callID)

Makros

FW_GETMSGHWND, FWFKT_GETMSGHWND(callback, varname)

Beschreibung

Die Funktion liefert das Handle des Meldungsfensters.

GetMainFrameHwnd - Handle des Hauptfensters ermitteln

Definition

HWND GetMainFrameHwnd(void* p_callID)

Makros

FW_GETMAINFRAMEHWND, FWFKT_GETMAINFRAMEHWND(callback, varname)

Beschreibung

Die Funktion liefert das Handle des Hauptfensters.

GetClientHwnd - Handle des Clientfensters ermitteln

Definition

HWND GetClientHwnd(void* p_callID)

Makros

FW_GETCLIENTHWND, FWFKT_GETCLIENTHWND(callback, varname)

Beschreibung

Die Funktion liefert das Handle des Clientfensters.

GetContainerHwnd - Handle des Container-Fensters ermitteln

Definition

HWND GetContainerHwnd(void* p_callID)

Makros

FW_GETCONTAINERHWND, FWFKT_GETCONTAINERHWND(callback, varname)

Beschreibung

Die Funktion liefert das Handle des Container-Fensters.

MsgWinActivate - Meldungsfenster aktivieren/deaktivieren

Definition

void MsgWinActivate(void* p_callID, bool on)

Makros

FW_MSGWINACTIVATE, FWFKT_MSGWINACTIVATE(callback, varname)

Beschreibung

Die Funktion aktiviert (`on==true`) oder deaktiviert (`on==false`) das Meldungsfenster. Durch das Deaktivieren wird das Meldungsfenster entfernt. Voraussetzung für das Deaktivieren ist das Vorhandensein der Funktion `FwownShowMsg()`. Diese Funktion ist bei deaktiviertem Meldungsfenster für das Verarbeiten der Meldungen verantwortlich. Bei deaktiviertem Meldungsfenster werden die Menüpunkte des Menüs *Anzeige/Meldungsfenster* deaktiviert. Durch das Aktivieren des Meldungsfensters wird das Fenster nicht direkt wieder sichtbar. Erst die Ausgabe der nächsten Meldung oder das explizite Anschalten über das Menü schalten das Meldungsfenster wieder ein.

FwownShowMsg - Meldung verarbeitenDefinition

```
FWOWN_API void CALLBACK FwownShowMsg(CallBackFkt p_callBack, void* p_callID, _TCHAR* p_msg);
```

Aufrufer

Standard-Framework

Aufrufzeitpunkt

Wenn eine Meldung ausgegeben werden soll und das Meldungsfenster deaktiviert ist.

Beschreibung

Die Funktion wird zur Verarbeitung einer Meldung aufgerufen, wenn das Meldungsfenster deaktiviert ist. Der Parameter `p_msg` zeigt auf die Meldung. Die Benutzerbibliothek hat somit zusammen mit der Funktion `MsgWinActivate()` die Möglichkeit, die Ausgabe der Meldungen selbst zu behandeln.

ShowMsg - Meldung im Meldungsfenster ausgebenDefinition

```
void ShowMsg(void* p_callID, _TCHAR* msg)
```

Makros

```
FW_MSGWINSHOWMSG, FWFKT_MSGWINSHOWMSG(callback, varname)
```

Beschreibung

Mit Hilfe dieser Funktion können Texte in das Meldungsfenster von FIX/Win geschrieben werden. Der Text ist als Parameter `msg` zu übergeben. Das Zeichen `'\n'` wird dabei als Zeilenumbruch interpretiert.

SetStatboxMsg - Meldung in Statuszeile ausgebenDefinition

```
BOOL SetStatboxMsg(void* p_callID, _TCHAR* msg)
```

Makros

```
FW_STATBOXSETMSG, FWFKT_STATBOXSETMSG(callback, varname)
```

Beschreibung

Mit dieser Funktion kann der Text der Statuszeile von FIX/Win festgelegt werden. Als Parameter `msg` ist der anzuzeigende Text mitzugeben. Die Funktion kehrt mit `TRUE` zurück, wenn die Statuszeile zum Zeitpunkt des Aufrufs sichtbar ist. Ansonsten wird `FALSE` zurückgeliefert. Der Text der Statuszeile wird von FIX/Win nach dem Verbindungsaufbau mit dem Programmnamen, dem Benutzernamen und dem Hostnamen belegt.

ShowMessageBox- nicht modale Meldung anzeigen

Definition

```
void ShowMessageBox(void* p_callID, _TCHAR* message, _TCHAR* title)
```

Makros

```
FW_SHOWMESSAGEBOX, FWFKT_SHOWMESSAGEBOX(callback, varname)
```

Beschreibung

Die Funktion zeigt die Meldung `message` in einer nicht modalen Meldungsbbox mit dem Titel `title` an. Als Vaterfenster wird das oberste Fenster in der Hierarchie verwendet. Bei dem Standard-Framework handelt es sich dabei um das Hauptfenster. Dieses Fenster wird beim Anzeigen der Meldungsbbox deaktiviert und beim Entfernen der Meldungsbbox wieder aktiviert. Auf diese Weise entsteht eine Art Pseudo-Modalität.

5 Spezielle Features von FIX/Win

1 Anzeige von modalen Dialogen in der Benutzerbibliothek

Wenn modale Dialoge innerhalb von bestimmten Funktionen der Benutzerbibliothek angezeigt werden, kann es zu einer Störung des Netzwerkverkehrs zwischen FIX/Win und der FIX-Anwendung kommen. Der Grund dafür liegt darin, dass modale Dialoge eine eigene Ereignis-Warteschlange starten und dass das Verarbeiten von Daten, die von der FIX-Anwendung kommen, über die Ereignis-Warteschlange gesteuert wird. Dadurch kommt es zu einem rekursiven Aufruf der Funktion, die die Daten der FIX-Anwendung verarbeitet. Diese hat jedoch noch nicht die Daten des vorherigen Aufrufs komplett abgearbeitet, da sie auf das Beenden des modalen Dialogs wartet.

Aus diesem Grund dürfen in Funktionen, die innerhalb der Abarbeitung der Daten der FIX-Anwendung aufgerufen werden, keine modalen Dialoge gestartet werden. Folgende Funktionen der Benutzerbibliothek sind davon betroffen:

- `FwownGetVersionInfo()`
- `FwownHook()` mit einem der Argumente `HOOK_PROTOINIT`, `HOOK_EXEC_PROGRAM`, `HOOK_EXEC_END`
- `FwownAddPaintArea()`
- `FwownUpdatePaintArea()`
- `FwownDelPaintArea()`
- `FwownExec()`

Als Lösung dieses Problems kann folgendes Verfahren verwendet werden:

- Statt des Aufrufs von `DialogBox()` ist der Dialog über `CreateDialog()` zu erzeugen. Als Wert für den Parameter `hWndParent` ist das Handle des Fensters anzugeben, das als Vater des Dialogs gelten soll (Das Handle kann z.B. mit `GetFixHwnd()` (siehe [GetFixHwnd - Handle des FIX/Win-Fensters ermitteln](#) auf Seite 91) oder mit `GetMainFrameHwnd()` (siehe [GetMainFrameHwnd - Handle des Hauptfensters ermitteln](#) auf Seite 94) ermittelt werden).
- Danach ist das Vaterfenster mittels `EnableWindow(hWndParent, FALSE)` zu deaktivieren.
- Jetzt kann der Dialog über `ShowWindow()` angezeigt werden.
- Beim Beenden des Dialogs ist das Vaterfenster über `EnableWindow(hWndParent, TRUE)` (in der Fensterprozedur des Dialogs) wieder zu aktivieren.

Durch dieses Verfahren entsteht eine Art *optische Modalität*. Das bedeutet, dass der Dialog sich im Vordergrund befindet und dass das Vaterfenster nicht bedient werden kann. Je nachdem, welches Fenster als Vaterfenster verwendet wird, ist eine teilweise Bedienung von FIX/Win möglich. Wird beispielsweise das FIX/Win Fenster verwendet, dann ist die Bedienung der FIX-Anwendung nicht mehr möglich. Das Hauptfenster kann jedoch noch bedient werden, indem es beispielsweise verschoben wird oder indem ein Menüpunkt ausgewählt wird.

Bezüglich des Programms FIX/Win besteht keine Modalität. Das bedeutet, dass die Funktion `ShowWindow()` sofort zurückkehrt, nachdem das Fenster angezeigt wurde. Damit entfällt die Möglichkeit, auf die Eingaben des Dialogs zu warten.

Ein häufig verwendeter Fall von modalen Dialogen sind Meldungsboxen, die über die Windows-Funktion `MessageBox()` gestartet werden. Als Ersatz hierfür kann die FIX/Win-Funktion `ShowMessageBox()` (siehe [ShowMessageBox-Box- nicht modale Meldung anzeigen](#) auf Seite 104) verwendet werden. Da es durch das oben beschriebene Verfahren jedoch dazu kommen kann, dass beim Auftreten von mehreren Meldungen mehrere Meldungsboxen gleichzeitig angezeigt werden, wird hier ein anderes Verfahren verwendet. Zur Lösung des Problems werden die Meldungen von `ShowMessageBox()` in eine Warteschlange geschrieben, die von einem eigenen Thread abgearbeitet wird. Deshalb kommt es nicht zu Störungen der Kommunikation mit der FIX-Anwendung und die Meldungen werden nacheinander angezeigt.

2 Verstecken des Hauptfensters von FIX/Win

Durch Angabe des Schalters

```
/hide
```

kann FIX/Win mit verstecktem Hauptfenster gestartet werden. Das Hauptfenster bleibt so lange unsichtbar, bis es von außen oder über die Benutzerbibliothek sichtbar gemacht wird.

Alternativ ist auch die Angabe der Option `SW_HIDE` als Wert für `lpStartupInfo.wShowWindow` beim Aufruf der Funktion `CreateProcess()` möglich. Dieser Wert wird auch gesetzt, wenn die Art des Fensters über die Verknüpfung definiert wird. Wird zusätzlich der Schalter `/hide` angegeben, dann überschreibt der Schalter die Einstellung in `lpStartupInfo`.

3 Ausgaben des Meldungsfensters umleiten

Es ist möglich, die Texte, die im FIX/Win Meldungsfenster ausgegeben werden sollen, abzufangen und in der Benutzerbibliothek zu verarbeiten.

Sämtliche im Text erwähnten Prototypen und Makros sind in der Datei `include/fwownStd.h` definiert. Da das Meldungsfenster Bestandteil des Standard-Frameworks ist, funktioniert die Änderung nur mit diesem Framework.

3.1 Abschalten des Meldungsfensters

Um eine Ausgabe im Meldungsfenster zu unterbinden, kann das Meldungsfenster von der Benutzerbibliothek abgeschaltet werden. Dazu ist FIX/Win-Funktion

```
void MsgWinActivate(void* p_callID, bool on)
```

aufzurufen. Der Parameter `on` bestimmt, ob das Meldungsfenster an- (`true`) oder ausgeschaltet (`false`) wird.

Das Abschalten des Meldungsfensters hat zur Folge dass:

- das Meldungsfenster versteckt wird, wenn es sichtbar ist.
- die Untermenüpunkte des Menüs Anzeige/Meldungsfenster grau dargestellt werden und nicht mehr ausgelöst werden können.
- bei einer Meldung keine Texte mehr in das Meldungsfenster oder in die durch `/log` angegebene Logdatei geschrieben werden.

Beim Anschalten des Meldungsfensters werden die Menüpunkte wieder normal dargestellt und sind auswählbar. Auch die Meldungen werden wieder in das Meldungsfenster (und in die Logdatei) geschrieben. Allerdings bleibt das Meldungsfenster unsichtbar und wird erst mit der nächsten Meldung nach dem Anschalten wieder angezeigt.

Um das Meldungsfenster komplett für FIX/Win abzuschalten, sollte der Aufruf von `MsgWinActivate()` relativ früh erfolgen. Ein guter Zeitpunkt ist das Senden des Hooks `HOOK_START`, da zu diesem Zeitpunkt noch keine Meldungen in das Fenster geschrieben wurden. (Nur bei fatalen Fehlern (z.B. bei zu wenig Speicher) kann es vorkommen, dass zu diesem Zeitpunkt schon Meldungen im Fenster stehen.)

3.2 Abfangen von Meldungen

Wenn das Meldungsfenster abgeschaltet ist, dann werden sämtliche Meldungen an die Funktion

```
FWOWN_API void CALLBACK FwownShowMsg(CallBackFct p_callBack, void* p_callID, _TCHAR* msg)
```

in der Benutzerbibliothek übergeben. Der Parameter `msg` enthält den Meldungstext. Der Meldungstext kann Zeilenumbrüche enthalten, die von einer Meldung des Backends stammen. Die Zeichen für den Zeilenumbruch sind je nach Backend UNIX- oder Windows konform.

Das Vorhandensein dieser Funktion ist die Voraussetzung für das Abschalten des Meldungsfensters. Ist sie nicht in der geladenen DLL vorhanden, dann funktioniert das Abschalten über `MsgWinActivate()` nicht und es wird eine entsprechende Fehlermeldung ausgegeben.

3.3 Beispiel

Die folgenden Codeteile stammen aus einem Beispiel einer Fwown-DLL.

```
#include "fwown.h"

MsgWinActivateFct MsgWinActivate = NULL;

#define CMDACT_MSGBOX _T("activate_msgbox")
#define CMDDACT_MSGBOX _T("deactivate_msgbox")

FWOWN_API char* CALLBACK FwownExec(CallBackFct CallBack, void* p_callID, _TCHAR* str)
{
    FWFKT_MSGWINACTIVATE(CallBack, MsgWinActivate);
    if (_tcsncmp(cmd, CMDACT_MSGBOX, _tcslen(CMDACT_MSGBOX))==0) {
        MsgWinActivate(p_callID, true);
        return 0;
    }
    else if (_tcsncmp(cmd, CMDDACT_MSGBOX, _tcslen(CMDDACT_MSGBOX))==0) {
        MsgWinActivate(p_callID, false);
        return 0;
    }
    _stprintf_s(errorbuf, ERRORBUFLEN, _T("Kommando <%s> unbekannt",commandstring));
    return errorbuf;
}
```

Beim Senden des Kommandos "activate_msgbox" wird das Meldungsfenster angeschaltet. Beim Senden von "deactivate_msgbox" wird es abgeschaltet.

```
ShowMessageBoxFct ShowMessageBox = NULL;

FWOWN_API void CALLBACK FwownShowMsg(CallBackFct p_callBack,
    void* p_callID, _TCHAR* msg)
{
    FWFCT_SHOWMESSAGEBOX(p_callBack, ShowMessageBox);
    ShowMessageBox(p_callID, msg, _T("FWOWN"));
}
```

Wenn das Meldungsfenster abgeschaltet wurde und eine Meldung ausgegeben wird, dann wird die Methode `FwownShowMsg()` aufgerufen. Diese Implementierung gibt den Meldungstext mit Hilfe der Funktion `ShowMessageBox()` als nicht modale Meldung aus.

4 Erstellen von Protokoll-Profilen

Protokoll-Profile sind hilfreich beim Ermitteln der Netzauslastung durch das Protokoll zwischen FIX und FIX/Win.

4.1 Funktionsweise des Protokolls

Um die Ausgabedatei zu verstehen, ist es notwendig zu wissen, wie das Protokoll zwischen FIX und FIX/Win funktioniert.

Zwischen FIX und FIX/Win bestehen zwei Kanäle, die auf FIX Seite stdout (stdin) und stderr zugeordnet werden. FIX/Win sendet auf dem Kanal stdout Tasten, die von FIX gelesen werden. Auf dem Kanal stderr werden von FIX/Win Codes gesendet, die als Signale bei FIX ankommen. FIX sendet auf dem Kanal stderr Fehlermeldungen, die in das Meldungsfenster von FIX/Win geschrieben werden. Auf dem Kanal stdout sendet FIX die Bildschirmausgaben in einem bestimmten Format. Da alle anderen Daten ein triviales Format haben und von der Datenmenge eher klein sind, werden bei Protokoll-Profilen nur die Bildschirmausgaben ausgewertet. (Ausnahme: Bei der Verwendung von `fx_transfer_to_frontend()` werden eventuell große Datenmengen von FIX/Win an FIX gesendet. Diese Daten werden als Tasten (Events) verpackt. Da die Menge der an FIX zurückgesendeten Daten jedoch in der eigenen Benutzerbibliothek bestimmt wird, ist sie leicht zu bestimmen und muss nicht unbedingt protokolliert werden.)

Aus Sicht von FIX werden Bildschirmausgaben einfach auf den Kanal stdout geschrieben. Eine einzelne Ausgabe besteht dabei aus einem Byte Opcode gefolgt von mehreren Bytes, die als Parameter verwendet werden. Danach kommt ein 0-Byte. FIX hat dabei wenig Einfluss darauf, wann diese Daten über das Netz an FIX/Win gesendet werden. Die C-Laufzeit Bibliothek sammelt die Daten, die auf stdout geschrieben werden, in einem internen Puffer und erzeugt beim Überlauf eine Ausgabe, die von dem rexec-Daemon gelesen wird und an FIX/Win über das Netzwerk weitergeleitet wird. An bestimmten Stellen wird von FIX die Funktion `fflush()` aufgerufen, um die Ausgabe des internen Puffers und so die Weiterleitung an FIX/Win zu erzwingen. FIX/Win liest alle Bytes, die von FIX gesendet werden, in einen Puffer. Dieser Puffer wird einer Funktion übergeben, die daraus die Opcodes zusammen mit den Parametern liest und anhand dieser Werte den internen Textbildschirm von FIX/Win aufbaut. Beim Interpretieren des Puffers kann es zu verschiedenen Situationen kommen, in denen die Funktion verlassen werden muss:

- Der Puffer ist leer und der Bildschirm wurde vollständig aktualisiert und muss dargestellt werden.
- Der Puffer ist leer und der Bildschirm wurde nicht vollständig aktualisiert.
- Der Puffer nicht ist leer und der Bildschirm wurde vollständig aktualisiert und muss dargestellt werden.

Auf diese Situationen muss die aufrufende Funktion reagieren und

- Die verarbeiteten Bytes aus dem Puffer löschen.
- Neue Bytes nachlesen.
- Den Bildschirm darstellen.

4.2 Einschalten von Protokoll-Profilen

Durch Angabe des Kommandozeilenparameters

```
/protoprof
```

beim Start von FIX/Win wird die Datei

```
protoprof.log
```

im Arbeitsverzeichnis von FIX/Win erzeugt. Da für alle Verbindungen nur eine Datei erzeugt wird, sollte nur mit einer gleichzeitigen Verbindung gearbeitet werden.

4.3 Aufbau der Datei

Die Datei `protoprof.log` ist eine Textdatei, deren Zeilen nach einem bestimmten Format aufgebaut sind. Die ersten 8 Zeichen einer Zeile enthalten ein Tag, das die Art der Zeile bestimmt. Danach folgt das Datum und die Uhrzeit. Als dritter Bestandteil der Zeile folgt ein beschreibender Text. Die drei Bestandteile einer Zeile sind durch " : " getrennt. Dieses Format erlaubt eine Weiterverarbeitung mit Werkzeugen wie `grep`, `sed` und `awk`.

Arten von Tags

START/END

Diese Tags markieren den Start und das Ende der Verbindung zu FIX.

NET

Dieses Tag markiert alle Einträge, die den internen Puffer betreffen. Bei

```
READ NETBUFFER
```

wurde eine Anzahl Bytes vom Netzwerk in den Puffer gelesen. Die Parameter beschreiben

- die Anzahl Bytes, die gelesen wurde.
- die Anzahl Bytes, die vom vorherigen Aufruf noch im Puffer waren.
- die Anzahl Bytes, die jetzt im Puffer sind.

Der Text

```
START TRANSLATE
```

wird unmittelbar vor dem Aufruf der Funktion ausgegeben, die den Puffer interpretiert. Sie erzeugt Ausgaben mit dem Tag `OPCO` (siehe weiter unten). Nach der Rückkehr der Funktion wird der Text

```
END TRANSLATE
```

zusammen mit der Anzahl Bytes, die interpretiert wurden, und der Anzahl, die übrig bleiben, ausgegeben.

STAT

Dieses Tag markiert alle Einträge einer Statistikausgabe. Jede Zeile enthält einen Opcode, die Anzahl der Vorkommen und die Summe der Bytes. Zusätzlich wird für den Netzwerkpuffer eine solche Zeile ausgegeben. Beim Lesen kommt es auch vor, dass 0 Bytes in den Puffer gelesen werden, weil keine weiteren Daten über das Netzwerk gesendet wurden. Solche Fälle werden bei der Statistik nicht berücksichtigt.

Die Werte der Statistik können von außen zurückgesetzt werden. In diesem Fall enthält die Datei die Meldung

```
Values initialised
```

zusammen mit dem Tag `STAT`.

OPCO

Die Zeilen, die mit diesem Tag beginnen, beschreiben eine einzelne Operation zusammen mit Parametern. Dabei beginnt jede Meldung mit einer Zahl, die die Anzahl Bytes darstellt, die für den Opcode zusammen mit den Parametern notwendig ist. Danach folgt durch Doppelpunkt getrennt der Opcode und durch einen weiteren Doppelpunkt getrennt der oder die Parameter.

Beispiel:

```
6:SCR_STRING: '(EN)' - 67
```

Die Operation schreibt den String "(EN)" in den Bildschirm. Dazu werden 6 Bytes benötigt. Der Parameter 67 ist die X-Position nach der Ausgabe.

Die folgende Tabelle enthält die wichtigsten Opcodes mit einer kurzen Beschreibung.

Opcode	Bedeutung
SCR_SET	Bildschirm auf FIX Modus stellen
SCR_RESET	Bildschirm in RAW Modus zurücksetzen

Opcode	Bedeutung
SCR_GETTERM	Terminal-Parameter lesen
SCR_GETKEY	Taste lesen
SCR_ICONLIST	Iconliste festlegen
SCR_ICONSET	Iconset festlegen
SCR_SET_ATTR	aktuelles Attribute setzen (fett, invers, ...)
SCR_STYLE	aktuellen Stil setzen (Feld, Text, Menü, ...)
SCR_CLR	alles löschen
SCR_CLRTOEOL	bis Zeilenende löschen
SCR_BEGIN_REFRESH	Anfang des Bildaufbaus
SCR_END_REFRESH	Ende des Bildaufbaus
SCR_SYNC	Zwischenstand des Bildschirms darstellen
SCR_POLL	Lebenszeichen anfordern
SCR_POS	Position bestimmen
SCR_RIGHT	eine Position rechts
SCR_UP	eine Position nach unten
SCR_SHOW_CARET	Cursor (Caret) anzeigen
SCR_SET_IMESTATUS	IME Status festlegen
SCR_SEND_ALLCLICKS	Mausklick auf allen Positionen senden
SCR_POS_STATUSLINE	Position in der Statuszeile festlegen
SCR_BEEP	Ton ausgeben
SCR_STRING	Zeichenkette ausgeben
SCR_CLAIM_RECT	Bereich für Objekt belegen
SCR_DECLARE	Objekt definieren
SCR_UNDECLARE	Objekt freigeben
SCR_PUSH_CONTEXT	Zustand sichern
SCR_POP_CONTEXT	Zustand wiederherstellen
SCR_DECLARE_PAINTAREA	Paintarea definieren
SCR_UNDECLARE_PAINTAREA	Paintarea freigeben
SCR_UPDATE_PAINTAREA	Paintarea aktualisieren
SCR_CLAIM_PAINTAREA	Bereich für Paintarea belegen
SCR_REDECLARE	Paintarea neu definieren
SCR_DISCARD	alle Eingaben verwerfen
SCR_SHOW_TOOLTIP	Tooltip anzeigen
SCR_EXEC_STRING	Zeichenkette an Benutzerbibliothek senden
SCR_TRANSFER	Binärdaten lesen
SCR_CONTEXT_MENU_START	markiert Start des Kontextmenüs
SCR_CONTEXT_MENU_ENTRY	Eintrag hinzufügen
SCR_CONTEXT_MENU_END	markiert Ende des Kontextmenüs
SCR_SET_ACTIVEOBJ	markiert Start der Liste aktiver Objekte
SCR_ADD_ACTIVEOBJ	Eintrag hinzufügen

Abgesehen von dem Parameter zu SCR_STRING sind die anderen Parameter nicht unbedingt von Bedeutung. Sie sind für interne Zwecke von NSI gedacht. In Einzelfällen können die Parameter bei NSI erfragt werden.

Der Parameter zu SCR_STRING stellt die Zeichenkette dar, die in den Bildschirm geschrieben wird. Anhand dieser Zeichenkette ist zu erkennen oder manchmal auch nur zu vermuten, um welchen Programmteil es sich handelt. Um diese Programmteile besser zu finden, besteht die Möglichkeit, selbst Meldungen in die Datei zu schreiben.

Weitere

Alle anderen Tags markieren Zeilen, die der Benutzer selbst über den Menüpunkt

Diagnose/Protokoll Meldung

in die Datei schreiben kann. Dazu wird eine Dialogbox eingeblendet, in der Tag und Meldung eingegeben werden können. Weiterhin können zwei Checkboxes angekreuzt werden, die bewirken, dass eine Statistik ausgegeben wird und dass die Werte der Statistik danach zurückgesetzt werden.

Beispiel

Es soll ermittelt werden, wieviele Bytes zur Darstellung der Auftragsmaske über das Netz gesendet werden.

Dazu wird FIX/Win mit den Schaltern

```
/protoprof /diag 0
```

gestartet. Danach wird im Menü bis unmittelbar vor die Auftragsmaske navigiert. Jetzt wird in FIX/Win der Menüpunkt

Diagnose/Protokoll Meldung

gestartet, um eine Meldung in die Datei `protoprof.log` zu schreiben. (Deshalb muss der Schalter `/diag` angegeben werden. Sonst ist das Menü "Diagnose" nicht vorhanden). Dazu werden beispielsweise folgende Werte eingegeben:

Tag: BEFORE

Text: Auftragsmaske

Zusätzlich wird die Checkbox "zurücksetzen" markiert. Dadurch werden alle Statistikwerte auf 0 gesetzt.

Dann wird die Auftragsbearbeitung betreten. Danach wird nochmals der Menüpunkt

Diagnose/Protokoll Meldung

in FIX/Win gestartet. Es werden folgende Werte eingegeben:

Tag: AFTER

Text: Auftragsmaske

Statistik: ja

zurücksetzen: ja

In der Datei `protoprof.log` kann jetzt anhand der Tags `AFTER` und `BEFORE` und anhand des Textes "Auftragsmaske" der Teil identifiziert werden, der zum Laden und Darstellen der Auftragsmaske gehört. Am Ende des Teils steht folgende Statistik:

```
STAT : 12/27/07 16:18:54 : =====
STAT : 12/27/07 16:18:54 : Operation          : Cnt   : Bytes
STAT : 12/27/07 16:18:54 : =====
STAT : 12/27/07 16:18:54 : SCR_GETKEY           :      2 :      4
STAT : 12/27/07 16:18:54 : SCR_ICONLIST        :      1 :      3
STAT : 12/27/07 16:18:54 : SCR_ICONSET         :      2 :     16
STAT : 12/27/07 16:18:54 : SCR_SET_ATTR        :    173 :    850
STAT : 12/27/07 16:18:54 : SCR_STYLE           :    114 :    342
STAT : 12/27/07 16:18:54 : SCR_BEGIN_REFRESH   :      4 :      8
STAT : 12/27/07 16:18:54 : SCR_END_REFRESH     :      4 :      8
STAT : 12/27/07 16:18:54 : SCR_SYNC            :      2 :      4
STAT : 12/27/07 16:18:54 : SCR_POS             :     78 :    312
STAT : 12/27/07 16:18:54 : SCR_RIGHT           :     40 :     80
STAT : 12/27/07 16:18:54 : SCR_SHOW_CARET     :      3 :      9
STAT : 12/27/07 16:18:54 : SCR_SET_IMESTATUS   :      1 :      3
STAT : 12/27/07 16:18:54 : SCR_POS_STATUSLINE :      2 :      8
STAT : 12/27/07 16:18:54 : SCR_STRING          :    275 :   2457
STAT : 12/27/07 16:18:54 : SCR_CLAIM_RECT      :     15 :    208
STAT : 12/27/07 16:18:54 : SCR_DECLARE         :      3 :     48
STAT : 12/27/07 16:18:54 : SCR_UNDECLARE       :      1 :      4
STAT : 12/27/07 16:18:54 : SCR_DECLARE_PAINTAREA :    21 :   1246
STAT : 12/27/07 16:18:54 : SCR_CLAIM_PAINTAREA :     42 :    628
```

```
STAT : 12/27/07 16:18:54 : SCR_SET_ACTIVEOBJ      :      1      :      3
STAT : 12/27/07 16:18:54 : SCR_ADD_ACTIVEOBJ      :      3      :     12
STAT : 12/27/07 16:18:54 : Netbuffer              :      4      :    6253
STAT : 12/27/07 16:18:54 : =====
```

Dazu lassen sich folgende Aussagen machen:

- Es wurden 4 Puffer vom Netz mit insgesamt 6253 Bytes gelesen.
- Den Hauptanteil hatte die Operation SCR_STRING mit 2457 Bytes.
- Auf Position 3 liegt SCR_DECLARE_PAINTAREA mit 1246 Bytes jedoch nur 21 Aufrufen. Das bedeutet, dass jeder Aufruf im Schnitt 60 Bytes belegt.

Hinweise

- Die Datei `protoprof.log` sollte mit einem Editor/Viewer gelesen werden, der UTF-8 kann. Sie enthält dazu die entsprechende Byte Order Mark.
- Die Datei `protoprof.log` wird bei jeder Schreiboperation geöffnet und danach wieder geschlossen. Alle Zeilen werden an die Datei hinten angehängt. Der Benutzer ist selbst dafür verantwortlich, die Datei zu entfernen.
- Bei manchen Arten von Operationen werden nicht alle Parameter ausgegeben, so dass anhand der Parameter nicht auf die Anzahl Bytes geschlossen werden kann.
- Verschiedene Längen für gleiche Operationen ergeben sich auf Grund der Tatsache, dass Zahlen als Zeichenketten übermittelt werden. So wird z.B. beim ersten Laden der Maske für das Objekt die ID 7 verwendet (Länge = 1 Byte) und beim zweiten Laden die ID 350 (Länge = 3)

4.4 Optimierungen

4.4.1 Refresh

Eine Optimierungsmöglichkeit, die von der Anwendung aus gesteuert werden kann, liegt darin, einige `refresh()`-Aufrufe zu unterbinden. FIX und FIX/Win unterscheiden zwei Situationen, in denen der aktuelle Bildschirm dargestellt werden soll:

- Auf dem Bildschirm hat sich etwas geändert, das der Benutzer sofort sehen soll. Diese Art wird SYNC-Refresh genannt.
- Vom Benutzer wird eine Taste erwartet. Deshalb wird vorher der aktuelle Bildschirm dargestellt. Diese Art wird NOSYNC-Refresh genannt.

Aktualisierungen der ersten Art können in vielen Fällen weggelassen werden, weil der Bildschirminhalt sich danach auf jeden Fall wieder ändert und die Zeit bis zum nächsten Bildschirmaufbau sehr kurz ist, so dass der Benutzer den geänderten Bildschirm oft nicht wahrnimmt.

Beispiel:

Der Benutzer steht auf einem Menüpunkt. Beim Auslösen wird eine Maske dargestellt, die sich über das Menü legt. Standardmäßig geht FIX wie folgt vor:

- Der Menüpunkt, der invertiert dargestellt wird, wird nochmals mit dem Attribut unterstrichen auf dem virtuellen Bildschirm ausgegeben.
- Damit der Benutzer diese Änderung sieht, wird der Bildschirm mittels eines SYNC-Refresh an FIX/Win übertragen und dargestellt.
- Danach wird die Maske geladen und auf dem virtuellen Bildschirm dargestellt.
- Durch das abschließende Anfordern einer Taste wird die Maske über einen NOSYNC-Refresh in FIX/Win dargestellt.

Die Darstellung durch den SYNC-Refresh im zweiten Schritt kann weggelassen werden. Auch wenn die Maske diesen Bereich nicht überdeckt, werden die Änderungen (Darstellen des Menüpunktes mit Attribut unterstrichen) beim NO-SYNC-Refresh in Schritt 4 übertragen und dargestellt.

Es können jedoch nicht alle SYNC-Refresh Aufrufe weggelassen werden. Zum Beispiel muss der Text "Holen" in der obersten Zeile einer Maske auf jeden Fall angezeigt werden. Das Lesen von Daten kann eine längere Operation sein, was dem Benutzer auf jeden Fall mitgeteilt werden sollte. Die Änderung wird auch nicht durch einen NOSYNC-Refresh angezeigt, weil erst dann wieder eine Taste eingelesen wird, wenn die Daten gelesen wurden. Zu diesem Zeitpunkt steht jedoch längst ein anderer Text in der Maske. Diese Tatsache macht es notwendig, dass das Abschalten von SYNC-Refreshs steuerbar ist.

Damit ein SYNC-Refresh überhaupt abgeschaltet werden kann, ist die FIX-Ressource (in `fix.rc`)

```
SkipSyncRefresh
```

auf den Wert `TRUE` zu setzen. Wird die Ressource nicht definiert oder besitzt die Ressource den Wert `FALSE`, dann verhält sich FIX wie bisher und führt alle SYNC-Refreshs aus.

Der zweite Schritt besteht darin in der Anwendung den Wert der Variablen

```
S_skip_sync_refresh
```

auf `TRUE` zu setzen. Besitzt diese Variable den Wert `FALSE`, dann werden alle SYNC-Refreshs ausgeführt. Damit besteht für die Anwendung die Möglichkeit, SYNC-Refreshs nur für bestimmte Abschnitte der Anwendung zu unterbinden.

Bisher wurde ein SYNC-Refresh durch Aufruf der Funktion `refresh()` in FIX und in der Anwendung ausgelöst. FIX ruft dazu jetzt die Funktion

```
void sync_refresh(char *dbgmsg, int force);
```

auf. Der Parameter `dbgmsg` enthält einen kurzen Text, der auf `stderr` ausgegeben wird, wenn die Anwendung mit dem Schalter `-dev` gestartet wurde und wenn der SYNC-Refresh aufgrund von `SkipSyncRefresh/S_skip_sync_refresh` unterbunden wurde. Mit Hilfe dieser Meldung kann man fehlende SYNC-Refreshs finden. Wenn der Parameter `force` mit `TRUE` übergeben wird, dann wird ein SYNC-Refresh auf jeden Fall ausgeführt - egal wie `SkipSyncRefresh/S_skip_sync_refresh` besetzt sind. Die Funktion, die den Text "Holen" in der obersten Maskenzeile darstellt nutzt beispielsweise diese Möglichkeit.

FIX ruft die Funktion `refresh()` nicht mehr auf. Sie ist jedoch noch aus Kompatibilitätsgründen in der Bibliothek vorhanden, damit nicht die komplette Anwendung umgestellt werden muss. Jedoch ruft sie intern die Funktion

```
sync_refresh("refresh", FALSE);
```

auf. Wenn also `SkipSyncRefresh/S_skip_sync_refresh` gesetzt werden, dann bleiben alle Aufrufe von `refresh()` ohne Wirkung. Ist dies an einzelnen Stellen nicht gewünscht, dann ist dort die Funktion `sync_refresh()` (mit `force=TRUE`) zu verwenden.

Beispiel:

Das obige Beispiel wurde nochmals gestartet. Allerdings wurden die Werte `SkipSyncRefresh/S_skip_sync_refresh` auf `TRUE` gesetzt, so dass SYNC-Refreshs unterbunden werden.

Jetzt sieht die Statistik wie folgt aus:

```
STAT : 12/28/07 10:14:24 : =====
STAT : 12/28/07 10:14:24 : Operation           : Cnt  : Bytes
STAT : 12/28/07 10:14:24 : =====
STAT : 12/28/07 10:14:24 : SCR_GETKEY           :    2  :    4
STAT : 12/28/07 10:14:24 : SCR_ICONLIST        :    1  :    3
STAT : 12/28/07 10:14:24 : SCR_ICONSET         :    2  :   16
STAT : 12/28/07 10:14:24 : SCR_SET_ATTR        :   165 :   804
STAT : 12/28/07 10:14:24 : SCR_STYLE           :   111 :   333
STAT : 12/28/07 10:14:24 : SCR_BEGIN_REFRESH   :    2  :    4
STAT : 12/28/07 10:14:24 : SCR_END_REFRESH     :    2  :    4
STAT : 12/28/07 10:14:24 : SCR_POS             :    75 :   300
STAT : 12/28/07 10:14:24 : SCR_RIGHT           :    40 :    80
STAT : 12/28/07 10:14:24 : SCR_SHOW_CARET     :    3  :    9
STAT : 12/28/07 10:14:24 : SCR_SET_IMESTATUS   :    1  :    3
STAT : 12/28/07 10:14:24 : SCR_POS_STATUSLINE  :    2  :    8
```

```

STAT : 12/28/07 10:14:24 : SCR_STRING           : 268 : 2421
STAT : 12/28/07 10:14:24 : SCR_CLAIM_RECT       : 15  : 208
STAT : 12/28/07 10:14:24 : SCR_DECLARE          : 3   : 48
STAT : 12/28/07 10:14:24 : SCR_UNDECLARE        : 1   : 4
STAT : 12/28/07 10:14:24 : SCR_DECLARE_PAINTAREA : 21  : 1246
STAT : 12/28/07 10:14:24 : SCR_CLAIM_PAINTAREA  : 42  : 628
STAT : 12/28/07 10:14:24 : SCR_SET_ACTIVEOBJ    : 1   : 3
STAT : 12/28/07 10:14:24 : SCR_ADD_ACTIVEOBJ    : 3   : 12
STAT : 12/28/07 10:14:24 : Netbuffer            : 2   : 6138
STAT : 12/28/07 10:14:24 : =====

```

Hier ist zu beobachten, dass das Darstellen der Auftragsmaske nun 115 Bytes weniger benötigt und das nur 2 statt 4 Puffer gelesen werden. Weiterhin ist zu sehen, dass die Operation SCR_SYNC nicht mehr vorkommt.

Wenn nach dem Lesen eines Satzes aus der Datenbank in die Maske eine Statistik ausgegeben wird, ist zu sehen, dass SCR_SYNC wieder auftaucht, da die Anzeige des Textes "Holen" einen SYNC-Refresh erfordert.

4.4.2 Paintarea

Zur Darstellung von Paintareas werden im Wesentlichen zwei Operationen benötigt:

- SCR_DECLARE_PAINTAREA - legt die Daten (Typ, Text, Tooltip, ...) der Paintarea fest.
- SCR_CLAIM_PAINTAREA - bestimmt das Rechteck, in dem die Paintarea dargestellt werden soll.

Die Darstellung von Paintareas erfolgt dann, wenn ein Objekt angezeigt wird. Beim Entfernen des Objekts vom Bildschirm werden auch die Paintareas entfernt. Da eine SCR_DECLARE_PAINTAREA Operation relativ viele Bytes benötigt, wird mit einem Paintarea-Cache gearbeitet (siehe FIX-Handbuch). Dieser Cache bewirkt, dass beim erneuten Betreten eines Objekts die Aufrufe von SCR_DECLARE_PAINTAREA unterbleiben und statt dessen mit den Werten gearbeitet wird, die bereits an FIX/Win gesendet wurden. Voraussetzung dafür sind jedoch:

- Der Cache muss groß genug sein, damit sich die Paintareas noch im Cache befinden. Die Größe wird durch `S_pa_max_tabs` auf FIX-Seite bestimmt (siehe Handbuch). Die Größe auf FIX/Win Seite passt sich der Anzahl aktuell im Cache befindlicher Paintareas an.
- Das Objekt darf vor dem erneuten Betreten nicht freigegeben und neu geladen werden. Ein vorhandener Cache-Eintrag wird an der Objekt-ID erkannt. Wenn eine Maske freigegeben und neu geladen wird, dann wird eine neue Objekt-ID vergeben. Es handelt sich aus Sicht von FIX um ein völlig anderes Objekt. Es besteht keine Möglichkeit, das Auffinden des Cache-Eintrags an anderen Daten, wie beispielsweise dem Namen mfo-Datei oder dem Namen der Maske, festzumachen. Hierbei handelt es sich nicht um eindeutige Kriterien für ein und das selbe Objekt. Weiterhin wird die Objekt-ID von FIX/Win intern verwendet, um ein Objekt zu identifizieren.

Daraus ergeben sich folgende Optimierungsmaßnahmen:

- Vergrößern des Paintarea-Caches.
- Häufig benutzte Objekte nicht freigeben.

4.4.3 Fullwidth-Zeichen

Fullwidth-Zeichen belegen 2 Zeichenzellen, die durch zwei spezielle Attribute markiert werden. Das erste Attribut `FIRSTFULLWIDTHCOL` markiert die erste Zelle und das Attribut `LASTFULLWIDTHCOL` markiert die zweite Zelle. Dadurch kommt es dazu, dass Fullwidth-Zeichen einzeln gesendet werden, wobei vorher jeweils das Attribut umgeschaltet wird.

Durch Setzen der FIX-Ressource (in `fix.rc`)

```
NoFullwidthAttr
```

auf `TRUE` kann die Verwendung von Attributen für Fullwidth-Zeichen im Protokoll unterbunden werden. FIX/Win berechnet in diesem Fall die Attribute selbst.

Obwohl im virtuellen Bildschirm für jedes Fullwidth-Zeichen 2 Zeichen (weil 2 Zeichenzellen belegt werden) stehen, wird nur ein Zeichen gesendet. FIX/Win konstruiert daraus wieder 2 Zeichen. Das steigert die Optimierung nochmals. Für ein Fullwidth-Zeichen werden somit nur 3 statt 6 Bytes (UTF-8 Darstellung) benötigt.

4.4.4 Weitere Optimierungen

Durch Setzen der FIX-Ressource (in `fix.rc`)

`loopOpt`

auf `TRUE` werden weitere Optimierungen eingeschaltet:

- Wenn eine Zeichenkette aus mehreren gleichen Zeichen besteht, die hintereinander stehen, werden nicht alle Zeichen gesendet. Statt dessen wird die Anzahl zusammen mit dem Zeichen als Operation `SCR_STRING_CNT` gesendet. Solche Zeichenketten entstehen häufig Zeichen, wenn Grafikzeichen für den Rahmen verwendet werden.
- Das Attribut wird als Byte gesendet, wenn es zwischen 0 und 255 liegt. Dazu wird die neue Operation `SCR_SET_ATTR_BIN` verwendet. Früher wurde es als Zeichenkette gesendet, die je nach Größe des Wertes 1-5 Bytes belegte. Dieses Verfahren wird auch bei gesetzter Ressource verwendet, wenn der Wert > 255 ist.
- `SCR_BEGIN_REFRESH` und `SCR_END_REFRESH` werden von FIX/Win zur Zeit nicht benötigt und werden deshalb weggelassen.

6 Fehlermeldungen

Fehlermeldungen von FIX/Win werden im Meldungsfenster dargestellt. Jede Meldung hat eine Kennzeichnung, z.B. NT100. Dabei stehen die ersten beiden Buchstaben für das Modul, aus dem die Meldung kommt. Die Zahl ist willkürlich gewählt.

Es gibt die folgenden Module:

- BM Bitmaps
- CN Verbindungsaufbau
- FX FIX bzw. FIX/Win allgemein
- IB IconBox
- IM Iconmanager
- KL Keylabels
- LG Logo
- NT Netzwerk
- OB Objekte
- PR Protokoll
- PT Drucken (Hardcopy)
- RS Ressourcen
- VC Virtueller Text-Bildschirm
- VG Virtueller Grafik-Bildschirm

Ein Benutzer muss nicht wissen, was die Module im einzelnen sind. Wenn man aber einen Fehler meldet, kann die Angabe der ausgegebenen Fehler mit genauer Fehlernummer beim Aufspüren des Fehlers sehr hilfreich sein.

Einige Fehler sind als interne Fehler bezeichnet. Bitte informieren Sie uns, wenn ein solcher Fehler auftritt.

1 BM - Bitmaps

BM100: Unbekanntes Grafikzeichen <Code>-<Code>

Die FIX-Anwendung versucht, ein Grafikzeichen auszugeben, das FIX/Win nicht bekannt ist. FIX/Win gibt in diesem Falle ein Fragezeichen aus. Die Bitmap für das Grafikzeichen muss im Unterverzeichnis `LookAndFeel-src` des Gruppenverzeichnisses angelegt werden. Danach müssen die Bitmaps gepackt werden.

Verweise

Verwendung von Semigrafikzeichen auf Seite 57 und *Packen von Bitmaps* auf Seite 60

Vermutliche Fehlerquelle

FIX/Win oder FIX-Anwendung

BM150: FATAL: Fehler beim Erzeugen der Bitmap Speicherfläche

Die Speicherfläche, die zum Laden der Bitmaps benötigt wird, kann nicht erstellt werden. Vermutlich sind nicht genügend Windows-Ressourcen vorhanden.

Verweise

[Voraussetzungen für den Einsatz auf Seite 9](#)

Vermutliche Fehlerquelle

FIX/Win oder Rechnerkonfiguration

BM170: Fehler beim Öffnen von Datei <Datei> <Fehlercode>

Die angegebene Datei konnte von der Bitmap-Laderoutine nicht geöffnet werden.

Verweise

[Aufbau eines Gruppenverzeichnisses auf Seite 41](#)

Vermutliche Fehlerquelle

Fehlende Datei, defekte Datei oder keine Leserechte.

BM180: Fehler bei Speicheranforderung in BitmapLoad

Es ist nicht genügend Speicher vorhanden, um ein Bitmap zu laden

Verweise

[Voraussetzungen für den Einsatz auf Seite 9](#)

Vermutliche Fehlerquelle

Rechnerkonfiguration.

BM190: Bitmap-Infodatei für Stil <nr> und Größe <nr> nicht gefunden

Die Bitmap-Infodatei (* .bif) für die angegebene Stil- Größenkombination konnte nicht geladen werden

Verweise

[Aufbau eines Gruppenverzeichnisses auf Seite 41](#)

Vermutliche Fehlerquelle

Fehlende Datei, defekte Datei oder keine Leserechte.

2 CN - Verbindungsaufbau

CN100: Programmtabelle <Name> kann nicht geöffnet werden.

Die Datei, die die Programmtabelle enthält, kann nicht gelesen werden. Ihr Name wird mit dem Schalter /ptab übergeben. Wird dieser Parameter nicht angegeben, dann wird die Datei `program.fix` aus dem Arbeitsverzeichnis-Hauptverzeichnis von FIX/Win geladen.

Verweise

[/ptab <Path> auf Seite 19](#)

Vermutliche Fehlerquelle

Datei mit Programmtabelle fehlerhaft
falscher Übergabeparameter

CN110: Programmtabelle <Name>: Zeile <Nummer> zu lang

Eine Zeile der Programmtabelle ist zu lang. Die maximale Zeilenlänge beträgt 160 Zeichen.

Verweise

[/ptab <Path> auf Seite 19](#)

Vermutliche Fehlerquelle

Datei mit Programmtabelle fehlerhaft

CN120: Programmtabelle <Name>: Fehlendes Komma in Zeile <Nummer>

Alle Elemente eines Eintrags der Programmtabelle müssen durch Komma getrennt sein.

Verweise

[/ptab <Path> auf Seite 19](#)

Vermutliche Fehlerquelle

Datei mit Programmtabelle fehlerhaft

CN240: Angegebener Programmname ist nicht in Liste

Über Kommandozeilenparameter /pgm wurde ein Programmname angegeben, der nicht in der Programmtabelle vorkommt.

Verweise

[/ptab <Path> auf Seite 19](#)

Vermutliche Fehlerquelle

Falsche Programmtabelle
Schreibfehler bei Programmname

3 DV - Developer

DV100: kann Datei für Screendump nicht öffnen

Vermutliche Fehlerquelle:

keine Schreibberechtigung auf die Datei

4 FX - FIX bzw. FIX/Win allgemein

FX100: Fehlerkanal: <Meldung>

Fehlermeldung der FIX-Anwendung, die auf den Fehlerkanal (stderr) geschrieben wurde.

Vermutliche Fehlerquelle

Startskript
FIX-Anwendung

FX110: Keine Benutzerbibliothek installiert

Es wurde ein Funktionsaufruf der Benutzerbibliothek abgesetzt, ohne dass eine Bibliothek installiert war. Zum Installieren einer Benutzerbibliothek muss ihr Name zusammen mit dem Schalter /fwown beim Start von FIX/Win angegeben werden.

Verweise

[/fwown <Path> auf Seite 20](#)

Vermutliche Fehlerquelle

Falsche Übergabeparameter beim Start von FIX/Win
FIX-Anwendung und FIX/Win wurden nicht aufeinander angepasst.
Aufruf von FwownLibExec() war nicht erfolgreich.

FX200: Fehler <nr> beim Laden von Tooltip Font <name> <größe>

Der Font mit dem angegebenen Namen und der angegebenen Größe konnte nicht geladen werden. Entweder ist er nicht installiert, oder die Angabe des Namens ist falsch.

Verweise

[Konfiguration von Tooltips auf Seite 87](#)

Vermutliche Fehlerquelle

Angabe des Fonts in den Ressourcen TooltipFont, TooltipFontSize_M, TooltipFontSize_L, TooltipFontSize_H

5 IB - IconBox

IB100: Iconlist <Nummer> nicht vorhanden

Die FIX-Anwendung verlangt eine Iconliste, die in der aktuellen Definitionsdatei nicht vorhanden ist. Zur Behebung ist in der Definitionsdatei eine Iconliste mit der entsprechenden ID zu vereinbaren.

Verweise

[Aufbau der Icondefinitionsdatei auf Seite 46](#)

Vermutliche Fehlerquelle

Icondefinitionsdatei
FIX-Anwendung

IB110: Iconset <Nummer> nicht vorhanden

Die FIX-Anwendung verlangt ein Iconset, das in der aktuellen Definitionsdatei nicht vorhanden ist. Zur Behebung ist in der Definitionsdatei ein Iconset mit der entsprechenden ID zu vereinbaren.

Verweise

[Aufbau der Icondefinitionsdatei auf Seite 46](#)

Vermutliche Fehlerquelle

Icondefinitionsdatei,
FIX-Anwendung

IB120: Icondefinitionsdatei <Name> kann nicht geöffnet werden

Die Datei mit den Definitionen der IconBox ist wahrscheinlich nicht vorhanden. Sie wird im Gruppenverzeichnis des entsprechenden Programmes erwartet.

Verweise

[Aufbau eines Gruppenverzeichnisses auf Seite 41](#)

Vermutliche Fehlerquelle

Icondefinitionsdatei

IB130: Fehler in Icondefinitionsdatei <Name>

Die Datei mit den Definitionen der IconBox ist fehlerhaft. Die IconBox kann in diesem Fall nicht verwendet werden.

Verweise

[Aufbau der Icondefinitionsdatei auf Seite 46](#)

Vermutliche Fehlerquelle

Icondefinitionsdatei

IB140: aktuelle Iconliste braucht <nr> Zeilen

Die Anzahl Zeilen für die IconBox reicht nicht aus, um die aktuelle Iconliste darzustellen. Durch die Angabe der Ressource `IconboxLines` kann die Anzahl Zeilen erhöht werden.

Verweise

[Einstellungen über Ressourcen auf Seite 50](#)

Vermutliche Fehlerquelle

Icondefinitionsdatei,
Ressourcendatei für IconBox

IB150: Zu kleiner Abstand in SPACE Definition: <num>

Die Angabe der SPACE Definition muss mindestens 4 Pixel definieren, wenn die Ressource `Like97` angegeben wird..

Verweise

[Aufbau der Icondefinitionsdatei auf Seite 46](#)

Vermutliche Fehlerquelle

Icondefinitionsdatei,
Ressourcendatei für IconBox

IB160: Iconbox Ressourcendatei für Stil <nr> nicht gefunden

Die Ressourcendatei mit den Einstellungen für den Stil <nr> konnte nicht gefunden werden.

Verweise

[Einstellungen über Ressourcen auf Seite 50](#)

[Aufbau eines Gruppenverzeichnisses auf Seite 41](#)

Vermutliche Fehlerquelle

Ressourcdatei für IconBox nicht vorhanden

IB170: Icon-Bitmap <Path> für Stil <nr> nicht gefunden

Die Bitmap für ein Icon im Stil <nr> konnte im Pfad <Path> nicht gefunden werden.

Verweise

[Konfiguration der IconBox auf Seite 45](#)

[Aufbau eines Gruppenverzeichnisses auf Seite 41](#)

Vermutliche Fehlerquelle

Bitmap nicht vorhanden

IB180: Rahmen-Bitmap <Path> für Stil <nr> nicht gefunden

Die Bitmap für den Rahmen eines Icons im Stil <nr> konnte im Pfad <Path> nicht gefunden werden.

Verweise

[Konfiguration der IconBox auf Seite 45](#)

[Aufbau eines Gruppenverzeichnisses auf Seite 41](#)

Vermutliche Fehlerquelle

Bitmap nicht vorhanden

6 IM - Iconman

IM100: Unbekanntes Schlüsselwort in Zeile <Nummer>

In der Icondefinitionsdatei ist ein Syntaxfehler aufgetreten. Es wird an dieser Stelle ein Schlüsselwort erwartet (ICONSET oder ICONLIST), jedoch steht in der Definitionsdatei keines.

Verweise

[Aufbau der Icondefinitionsdatei auf Seite 46](#)

Vermutliche Fehlerquelle

Icondefinitionsdatei

IM110: Zu viele Iconlisten (max <Anzahl>) in Zeile <Nummer>

Die Anzahl der maximal zulässigen Iconlist-Definitionen wurde überschritten.

Verweise

[Aufbau der Icondefinitionsdatei auf Seite 46](#)

Vermutliche Fehlerquelle

Icondefinitionsdatei

IM130: Tabellenüberlauf der Iconlisten (max <Anzahl>) in Zeile <Nummer>

Die Anzahl der Iconeinträge (Icons und Spaces) aller Iconlisten ist auf einen Maximalwert begrenzt. Dieser Wert wurde beim Anlegen einer Iconliste überschritten.

Verweise[Aufbau der Icondefinitionsdatei auf Seite 46](#)Vermutliche Fehlerquelle

Icondefinitionsdatei

IM150: Nicht erlaubte Icon-ID <ID> in Zeile <Nummer>

Der Wert darf nicht als Icon-ID verwendet werden.

Verweise[Aufbau der Icondefinitionsdatei auf Seite 46](#)Vermutliche Fehlerquelle

Icondefinitionsdatei

IM170: Dateiname des Bitmaps zu lang in Zeile <Nummer>

Der Dateiname, der das anzuzeigende Bitmap angibt, ist länger als 12 Zeichen.

Verweise[Aufbau der Icondefinitionsdatei auf Seite 46](#)Vermutliche Fehlerquelle

Icondefinitionsdatei

IM180: Hinweistext zu lang in Zeile <Nummer>

Der Hinweistext zu einem Icon darf maximal 80 Zeichen (1 Zeile) lang sein.

Verweise[Aufbau der Icondefinitionsdatei auf Seite 46](#)Vermutliche Fehlerquelle

Icondefinitionsdatei

IM190: Zu viele Iconsets (max <Anzahl>) in Zeile <Nummer>

Die Anzahl der Iconsets übersteigt einen Maximalwert.

Verweise[Aufbau der Icondefinitionsdatei auf Seite 46](#)Vermutliche Fehlerquelle

Icondefinitionsdatei

IM200: Tabellenüberlauf Iconset (max <Anzahl>) in Zeile <Nummer>

Die Gesamtanzahl aller Iconsets übersteigt einen Maximalwert.

Verweise

[Aufbau der Icondefinitionsdatei auf Seite 46](#)

Vermutliche Fehlerquelle

Icondefinitionsdatei

IM210: Fehlendes 'END' in Zeile <Nummer>

Es wird das Schlüsselwort END in der Icondefinitionsdatei erwartet.

Verweise

[Aufbau der Icondefinitionsdatei auf Seite 46](#)

Vermutliche Fehlerquelle

Icondefinitionsdatei

IM220: Fehlendes ';' in Zeile <Nummer>

Es wird ein ';' an dieser Stelle in der Icondefinitionsdatei erwartet.

Verweise

[Aufbau der Icondefinitionsdatei auf Seite 46](#)

Vermutliche Fehlerquelle

Icondefinitionsdatei

IM300: <name> nicht mehr benutzbar, muss das Parsen abbrechen, Zeile <nr>

Fataler Fehler beim Einlesen der Icondefinitionsdatei. Oft Folgefehler von anderen Fehlern.

Verweise

[Aufbau der Icondefinitionsdatei auf Seite 46](#)

Vermutliche Fehlerquelle

Icondefinitionsdatei

IM301: IconListTab Record konnte nicht allokiert werden, Parsen abgebrochen, Zeile <nr>

Fehler bei der Speicheranforderung beim Einlesen der Icondefinitionsdatei.

Verweise

[Aufbau der Icondefinitionsdatei auf Seite 46](#)

Vermutliche Fehlerquelle

Icondefinitionsdatei,
Speichermangel

7 KL - Keylabels (Tasten-Label)

KL100: Unbekanntes FIX-Event <Name>

Die FIX-Anwendung will ein Tasten-Label ausgeben, welches FIX/Win nicht kennt. FIX/Win stellt in diesem Fall Fragezeichen als Dummy-Label dar.

Verweise

[Erstellen einer eigenen Tastenbelegung auf Seite 53](#)

Vermutliche Fehlerquelle

FIX/Win oder FIX-Anwendung

KL110: FIX-Event <Name> nicht mit Taste belegt

Die FIX-Anwendung stellt ein Tasten-Label dar, dem keine Windows-Taste und somit auch kein Bitmap zugeordnet ist. FIX/Win stellt in diesem Fall Fragezeichen als Dummy-Label dar.

Verweise

[Erstellen einer eigenen Tastenbelegung auf Seite 53](#)

Vermutliche Fehlerquelle

Tastenbelegungsdatei
FIX-Anwendung

KL120: Bitmap für Tasten-Label <Name> nicht vorhanden

Ein Tasten-Label kann nicht gezeichnet werden, weil das entsprechende Bitmap nicht vorhanden ist. In diesem Fall werden Fragezeichen als Dummy-Label gezeichnet.

Verweise

[Erstellen einer eigenen Tastenbelegung auf Seite 53](#)

Vermutliche Fehlerquelle

Fehlende oder fehlerhafte Bitmap-Datei

KL130: Unbekanntes Tasten-Label <Name>

Es wurde eine (Funktions) Taste gedrückt, der kein FIX-Event zugeordnet ist.

Verweise

[Erstellen einer eigenen Tastenbelegung auf Seite 53](#)

Vermutliche Fehlerquelle

Tastenbelegungsdatei

KL140: Tastenbelegungsdatei <Name> kann nicht geöffnet werden

Eine Datei mit Tastendefinitionen kann nicht geöffnet werden. Vermutlich ist sie nicht vorhanden.

Verweise

[Erstellen einer eigenen Tastenbelegung auf Seite 53](#)

Vermutliche Fehlerquelle

Datei `stdkey.fix` existiert nicht

KL150: Tastenbelegungsdatei <Name>: Zeile <Nummer> zu lang

Die Zeile einer Tastendefinition ist zu lang. Unter FIX/Win ist die Zeilenlänge auf 160 Zeichen begrenzt. In diesem Fall wird der Rest der Zeile einfach abgeschnitten.

Verweise

[Erstellen einer eigenen Tastenbelegung auf Seite 53](#)

Vermutliche Fehlerquelle

Tastenbelegungsdatei

KL160: Tastenbelegungsdatei <Name>: unbekannter Tastenname:<Name> in Zeile <Nummer>

Für eine Taste wurde ein unzulässiger Name verwendet.

Verweise

[Erstellen einer eigenen Tastenbelegung auf Seite 53](#)

Vermutliche Fehlerquelle

Tastenbelegungsdatei

KL170: Tastenbelegungsdatei <Name>: unbekanntes FIX-Event:<Name> in Zeile <Nummer>

Für ein FIX-Event wurde ein unzulässiger Name verwendet.

Verweise

[Erstellen einer eigenen Tastenbelegung auf Seite 53](#)

Vermutliche Fehlerquelle

Tastenbelegungsdatei

KL180: FATAL: Fehler beim Erzeugen der Datenstruktur für Tasten-Label

Die Datenstruktur zur Verwaltung der Tasten-Labels konnte nicht erzeugt werden.

Verweise

[Erstellen einer eigenen Tastenbelegung auf Seite 53](#)

Vermutliche Fehlerquelle

zu wenig Windows-Ressourcen
zu wenig Speicher

KL210: Rahmen-Bitmap <name> für Tasten-Labels in Stil <nr> und Größe <nr> nicht gefunden

Die Bitmap mit dem Namen <name> konnte für die angegebene Stil-/Größenkombination nicht gefunden werden.

Verweise

[Erstellen einer eigenen Tastenbelegung auf Seite 53](#)

Vermutliche Fehlerquelle

Datei nicht vorhanden

KL220: Bitmap-Infodatei <name> für Tasten-Labels in Stil <nr> und Größe <nr> nicht gefunden

Die Bitmap-Infodatei (*.bif) mit dem Namen <name> konnte für die angegebene Stil-/Größenkombination nicht gefunden werden.

Verweise

[Erstellen einer eigenen Tastenbelegung auf Seite 53](#)

Vermutliche Fehlerquelle

Datei nicht vorhanden

8 LG - Logo

LG100: ungültiger Wert der Ressource 'LogoCenterMode': <Wert>

Der angegebene Wert ist nicht zulässig

Verweise

[Anzeige eines Logos und Abspielen eines Tons auf Seite 85](#)

Vermutliche Fehlerquelle

Fehlerhafte Angabe

LG110: Fenster für Logo kann nicht erzeugt werden

Das Fenster zur Anzeige eines Logos kann nicht erzeugt werden.

Verweise

[Anzeige eines Logos und Abspielen eines Tons auf Seite 85](#)

Vermutliche Fehlerquelle

Zu wenig System-Ressourcen

LG120: Datei <name> für Logo kann nicht geladen werden

Die Bitmapdatei zur Darstellung eines Logos kann nicht geladen werden..

Verweise

[Anzeige eines Logos und Abspielen eines Tons auf Seite 85](#)

Vermutliche Fehlerquelle

Datei nicht vorhanden

9 NT - Netzwerk

NT100: Netzfehler beim Senden (NetSendStr) - Fehlercode: <Code>

Daten können nicht über das Netz zum Applikationsserver gesendet werden. Tritt dieser Fehler nur manchmal auf, ist das Netzwerk wahrscheinlich instabil.

Vermutliche Fehlerquelle

Netzverbindung

NT110: Service rexec unbekannt

Der Service rexec muss der Netzwerksoftware bekannt sein.

Vermutliche Fehlerquelle

Eintrag in der Datei `services` auf Windows-Seite

Der Service "exec" muss der Netzwerksoftware bekannt sein. Dies wird auf UNIX Rechnern meistens dadurch erreicht, dass ein Eintrag in der Datei `/etc/services` existiert, etwa:

`exec512/tcp`

Achtung: Die Art und Weise, wie dem System ein Service mitgeteilt wird, kann von Rechner zu Rechner verschieden sein. Konsultieren Sie Ihre System-Dokumentation unter "services" oder "inetd" oder fragen Sie Ihren Systemverwalter.

NT111: Hostname <Name> unbekannt - Fehlercode: <Code>

Der Name des Hosts, auf dem Anwendung gestartet werden soll, muss der Netzwerksoftware bekannt sein. Der Hostname kann mindestens über 3 Methoden, oder über eine Folge dieser Methoden, nachgesehen werden:

- Nameserver
- NIS, Network Information Services, früher bekannt als Yellow Pages
- die Systemdatei, auf der Hostnamen eingetragen werden, z.B. `/etc/hosts`. Der Name der Datei ist abhängig von der verwendeten TCP/IP Software.

Es hängt von der Konfiguration des jeweiligen Systems ab, welche dieser Methoden benutzt wird.

Die Fehlermeldung deutet jedenfalls darauf hin, dass der Hostname über die benutzten Methoden nicht bestimmt werden konnte.

Vermutliche Fehlerquelle

Eintrag in der Datei `hosts` auf Windows-Seite
Netzwerkconfiguration

NT112: Socket kann nicht erzeugt werden - Fehlercode <Code>

NT113: Verbindung kann nicht hergestellt werden - Fehlercode <Code>

NT114: Socket kann nicht gebunden werden - Fehlercode <Code>

NT115: Socketname kann nicht ermittelt werden - Fehlercode <Code>

NT116: Fehler im Select - Fehlercode <Code>

NT117: Fehler im Accept - Fehlercode <Code>

NT120: Fehler im AsyncSelect Datensocket - Fehlercode <Code>

NT121: Fehler im AsyncSelect Fehlersocket - Fehlercode <Code>

Diese Fehler treten auf, wenn die Netzwerksoftware nicht mit FIX/Win zusammenarbeitet. Weiterhin kann der Fehler auch in einer nicht zur Hardware passenden Netzsoftware liegen.

Vermutliche Fehlerquelle

Netzhardware

NT118: Server antwortet nicht

Anscheinend ist der Hostrechner, auf welchem die Anwendung gestartet werden soll, nicht betriebsbereit oder nicht richtig installiert.

Vermutliche Fehlerquelle

Hostrechner

NT119: Server Fehler: <Fehlermeldung>

Fehlermeldung des Servers, die angibt, weshalb die Verbindung nicht zustande kommt.

Vermutliche Fehlerquelle

Startskript oder FIX-Anwendung

NT140: Fehler beim Ermitteln der Funktion <name>- Win32 Fehlercode <nr>

FIX/Win wird von einem RemoteDesktop bedient. Statt der Adresse des Terminal-Servers wird die Adresse des RemoteDesktops an FIX gesendet. Dazu sind Funktionen aus dem TerminalServer-API notwendig. Diese Fehlermeldung deutet auf einen Fehler beim Ermitteln dieser Funktionen hin.

Vermutliche Fehlerquelle

Fehlende oder fehlerhafte Bibliothek mit TerminalServer-API

NT150: Fehler in Funktion WTSQuerySessionInfo - Win32 Fehlercode <nr> - IP-Adresse des Servers wird benutzt

FIX/Win wird von einem RemoteDesktop bedient. Statt der Adresse des Terminal-Servers wird die Adresse des RemoteDesktops an FIX gesendet. Beim Ermitteln dieser Adresse ist ein Fehler aufgetreten. Deshalb wird statt dieser Adresse die Adresse des Terminal-Servers verwendet, was zu Problemen bei der Lizenzierung führen kann.

Vermutliche Fehlerquelle

Fehlerhafte Bibliothek mit TerminalServer-API

NT160: Alle Ports belegt (Bereich: <von> - <bis>)

Der angegebene Bereich wurde als TCP-IP Ports für FIX/Win definiert. Alle Ports in diesem Bereich sind belegt. Es kann keine Verbindung aufgebaut werden

Verweise

[Vorgabe von Ports](#) auf Seite 85.

Vermutliche Fehlerquelle

Zu viele Verbindungen oder falsche Einstellung in `FirstPort`, `LastPort`

10 OB - Objekte

OB150: FATAL: Unzulässige Farbangabe: <Wert>

Der angegebene Wert kann nicht in eine Farbe umgesetzt werden.

Verweise

[Erstellen eigener Farbzusammenstellungen](#) auf Seite 63

Vermutliche Fehlerquelle

Farbangabe in Farbzusammenstellungstabelle `coltab.fix`

OB160: Fontdatei für Stil <nr> und Größe <nr> nicht gefunden

Für die angegebene Stil-/Größenkombination wurde keine Fontdatei gefunden.

Verweise

[Anpassung von Schriftarten auf Seite 67](#)

Vermutliche Fehlerquelle

Datei existiert nicht.

OB170: Caret-Bitmap für Stil <nr> und Größe <nr> nicht gefunden

Für die angegebene Stil-/Größenkombination wurde keine Bitmap zur Darstellung der Schreibmarke gefunden.

Verweise

[Anpassung der Schreibmarke auf Seite 74](#)

Vermutliche Fehlerquelle

Datei existiert nicht.

11 PR - Protokoll

PR100: Fehler beim Öffnen von Ausgabedatei <Name>

Die Datei, die die Ausgabe aufnimmt, welche die FIX-Anwendung auf die Standardausgabe schreibt, lässt sich nicht zum Schreiben öffnen. Wahrscheinlich wurde ein falscher Dateiname festgelegt.

Verweise

[Ausgabe von Listen auf Seite 39](#)

Vermutliche Fehlerquelle

Angabe in Dialogbox *Einstellungen*

PR110: Fehler <Nummer> beim Ausführen von <Programm>

Das externe Programm zum Betrachten einer Ausgabedatei kann nicht gestartet werden.

Vermutliche Fehlerquelle

Angabe in Dialogbox *Einstellungen*

PR120: FATAL: Unbekannter Rückgabewert:<Nummer> von 'Translate'

Interner Fehler von FIX/Win.

PR130: Fehlermeldung in der Benutzerbibliothek: <Meldung>

Von der Benutzerbibliothek wurde ein Fehler zurückgeliefert, der zusammen mit dieser Fehlermeldung ausgegeben wird.

Verweise

[Erstellen einer Benutzerbibliothek auf Seite 89](#)

Vermutliche Fehlerquelle

Funktionen der Benutzerbibliothek

PR140: Fatal: Zu viele Objekt-Tabellen

Die Anzahl der gleichzeitig aktiven FIX-Objekte wurde überschritten

Vermutliche Fehlerquelle

FIX-Anwendung

PR150: Fehler: Unbekannter Protokoll-Code <nr> (Puffer=<text>)

FIX/Win hat einen Protokoll-Code empfangen, der nicht verarbeitet werden kann.

Vermutliche Fehlerquelle

Fataler Fehler, Falsche FIX Version

12 PT - Drucken

PT110: Fehler beim Drucken in Funktion <name> - Win32 Fehlercode <nr>

Interner Fehler beim Drucken in der Funktion <name>

PT120: Fehler beim Drucken Ermitteln des Device Contextes

Interner Fehler beim Drucken

PT130: Die Hardcopy passt nicht auf die Seite

Eine Hardcopy passt nicht mehr auf eine Seite.

Vermutliche Fehlerquelle

Zu kleine Seitengröße definiert oder interner Fehler

13 RS - Ressourcen

RS180: Datei <name> ist nicht lesbar

Die Ressourcdatei <name> kann nicht gelesen werden.

Vermutliche Fehlerquelle

Nicht vorhandene Datei oder keine Leserechte

RS190: <Dateiname>, Zeile <nr>: Zeile zu lang

Eine Zeile der Datei ist zu lang. Es sind maximal 512 Zeichen erlaubt.

RS200: eingeschlossen von <name>, Zeile <nr>

In einer Ressourcdatei, die über `#include` eingelesen wurde, ist ein Fehler aufgetreten, der bereits gemeldet wurde. Die Meldung RS200 gibt die Datei und die Zeilennummer der `#include` Anweisung an.

RS210: in Datei <name>, Zeile <nr>

In einer Ressourcdatei ist ein Fehler aufgetreten, der bereits gemeldet wurde. Die Meldung RS210 gibt die Datei und die Zeilennummer an.

14 VC - Virtueller Text-Bildschirm

VC100: FATAL: Virtueller Bildschirm nicht initialisiert

Interner Fehler von FIX/Win

Vermutliche Fehlerquelle

FIX/Win

VC110: FATAL: unbekanntes Style-Attribut (WP_) bei VcbRefresh()

Interner Fehler. Eine Position des Bildschirms kann bezüglich des Attributes nicht korrekt ausgegeben werden.

Vermutliche Fehlerquelle

FIX/Win oder FIX

VC120: FATAL: Fehler beim Erzeugen des virtuellen Text-Bildschirms

Der virtuelle Bildschirm zur Verwaltung des Bildschirm der FIX-Anwendung konnte nicht erzeugt werden. Wahrscheinlich ist dazu nicht mehr genügend Speicher vorhanden.

Vermutliche Fehlerquelle

zu wenig Hauptspeicher
zu wenig Windows-Ressourcen

15 VG - Virtueller Grafik-Bildschirm

VG100: FATAL: Fehler beim Erzeugen des virtuellen Grafik-Bildschirms

Der virtuelle Bildschirm zur Verwaltung des FIX/Win-Bildschirms konnte nicht erzeugt werden. Wahrscheinlich ist dazu nicht mehr genügend Speicher vorhanden.

Vermutliche Fehlerquelle

zuwenig Hauptspeicher
zuwenig Windows-Ressourcen

VG110: Farbzusordnungsdatei <Name> nicht lesbar

Die Farbzusordnungsdatei kann nicht zum Lesen geöffnet werden. Wahrscheinlich ist die Datei nicht vorhanden.

Verweise

[Erstellen eigener Farbzusordnungsstabellen auf Seite 63](#)

Vermutliche Fehlerquelle

Fehlende oder defekte Farbzuordnungsdatei

VG120: Farbzuordnungsdatei Stil <Nummer>: Zeile <Nummer> zu lang.

Die Zeile der Farbzuordnungsdatei des Stils <Nummer> enthält eine Zeile, die zu lang ist.

Verweise

[Erstellen eigener Farbzuordnungstabellen auf Seite 63](#)

Vermutliche Fehlerquelle

Farbzuordnungsdatei

VG130: Farbzuordnungsdatei Stil <Nummer>: Unbekannter Farb- oder Elementname in Zeile <Nummer>

In Zeile <Nummer> der Farbzuordnungsdatei ist entweder der Elementname oder ein Farbname unbekannt.

Verweise

[Erstellen eigener Farbzuordnungstabellen auf Seite 63](#)

Vermutliche Fehlerquelle

Farbzuordnungsdatei

VG160: Farbzuordnungsdatei Stil <Nummer>: Fehlender Farbname in Zeile <Nummer>

Zu einem Element fehlt eine Farbangabe. Zu jedem der Elemente muss eine Hinter- und Vordergrundfarbe angegeben werden.

Verweise

[Erstellen eigener Farbzuordnungstabellen auf Seite 63](#)

Vermutliche Fehlerquelle

Farbzuordnungsdatei

VG170: Fehler in Funktion <name> beim Druck des Bildschirm - Win32 Fehlercode: <nr>

Beim Drucken ist ein interner Fehler aufgetreten

VG180: Farbzuordnungsdatei Style <nr>: nicht vorhanden

Die Farbzuordnungsdatei für den Stil <nr> ist nicht vorhanden.

Verweise

[Erstellen eigener Farbzuordnungstabellen auf Seite 63](#)

Vermutliche Fehlerquelle

Farbzuordnungsdatei fehlt

7 Migrationshinweise

Dieses Dokument enthält Hinweise zu den Änderungen der aktuellen FIX/Win Version. Damit soll es möglich sein, Änderungen und Anpassungen von einer alten Version in die neue Version zu übernehmen. Die folgenden Abschnitte beziehen sich auf Änderungen, die in der Version 3.1.0 von FIX/Win vorgenommen wurden.

Die Version 3.1.0 wurde erstellt, um den Unicode-Zeichensatz in FIX-Anwendungen zu nutzen. Daher wurden sowohl FIX als auch FIX/Win auf die Verwendung des Unicode-Zeichensatzes angepasst. Die Anpassungen beschränken sich auf die *Basic Multilingual Plane*, die die ersten 65536 Zeichen von Unicode enthält und damit die Zeichen der meisten gebräuchlichen Sprachen unterstützt. Grund für diese Beschränkung ist im Wesentlichen die Verwendung des Datentyps `wchar_t` und der Funktionen, die diesen Datentyp verwenden. Unter Windows kann dieser Datentyp maximal 65536 Zeichen kodieren.

Abgesehen davon sind die Änderungen auf den Unicode-Zeichensatz beschränkt. Der Begriff "Unicode" definiert wesentlich mehr als den Zeichensatz, wie zum Beispiel die Schreibrichtung und die Verwendung von Akzent-Zeichen. Diese Dinge wurden jedoch weder in FIX/Win noch in FIX implementiert.

Allerdings beachten FIX und FIX/Win die Besonderheit, dass es bei der Darstellung von chinesischen Zeichen halb-breite und volle Zeichen gibt: Halfwidth- und Fullwidth-Zeichen. Bei dem zeichenzellenorientierten Bildaufbau von FIX belegt ein Halfwidth-Zeichen eine Zeichenzelle und ein Fullwidth-Zeichen belegt zwei Zeichenzellen.

1 Allgemeines

Der Featurelevel von FIX und von FIX/Win wurde von 4 auf 5 erhöht. Durch die Umstellung des Protokolls auf Unicode ist es notwendig, dass ein FIX-Programm, das mit der Version 3.1.0 gebunden wurde auch mit FIX/Win ab der Version 3.1.0 bedient wird. Umgekehrt ist es nicht möglich, mit FIX/Win Version 3.1.0 FIX-Programme zu bedienen, die mit FIX vor der Version 3.1.0 gebunden wurden.

FIX/Win wird nur als Unicode-fähige Version ausgeliefert, die intern für Zeichen den Typ `wchar_t` verwendet. Mit dieser Version lassen sich folgende FIX-Programme bedienen:

- Programme, die mit FIX 3.1.0 übersetzt wurden und einen der (8-Bit) Zeichensätze ISO-2, ISO-15, IBM437 oder GERMAN7 verwenden.
- Programme die mit FIX 3.1.0 Unicode übersetzt wurden und den Zeichensatz UTF-8 verwenden.

Damit FIX/Win mit Programmen zusammenarbeiten kann, die einen 8-Bit-Zeichensatz verwenden, setzt FIX unmittelbar vor der Kommunikation mit FIX/Win die 8-Bit-Zeichen nach UTF-8 um.

2 Zeichensatz der Konfigurationsdateien

Als Zeichensatz und Kodierung für die Konfigurationsdateien wird jetzt UTF-8 verwendet. Es ist jedoch möglich, das alte Format einzulesen. Nähere Hinweise sind im Abschnitt *Kodierung der Konfigurationsdateien* auf Seite 41 zu finden.

3 Ressourcedatei mit globalen Einstellungen

In der Datei

```
config/fixwin.frc
```

können globale Einstellungen vorgenommen werden, die FIX/Win betreffen und sich nicht auf eine bestimmte Anwendung beziehen. Eine Beschreibung der Einstellungen ist im Abschnitt [Globale Einstellungen von FIX/Win auf Seite 21](#) zu finden.

4 Zeichensatz von stderr und stdout

Der Zeichensatz, den FIX/Win für die Zeichen, die von der Anwendung auf stderr gesendet werden, annimmt, ist einstellbar. Eine Beschreibung ist im Abschnitt [Zeichensatz von stderr und stdout](#) auf Seite 86 zu finden.

5 Länderspezifische Einstellungen

Der Lademechanismus, den FIX/Win verwendet, um eine Windows-Ressource zu laden, wurde in der Version 3.1.0 überarbeitet. Welche Ressource geladen wird, ist jetzt vom Gebietschema abhängig. Eine Beschreibung dazu ist im Abschnitt [Anpassen von Meldungen, Menüpunkten und Dialogen](#) auf Seite 74 zu finden.

Weiterhin wird bei dem Gebietsschema "Japanisch" die Schriftart "MS Gothic" verwendet (siehe [Anpassung von Schriftarten](#) auf Seite 67).

6 Verwenden von Schriftarten des Windowssystems

Ab der Version 3.1.0 ist es möglich, Schriftarten des Windowssystems zu verwenden, die den Unicode-Zeichensatz unterstützen. Dazu wurde die Datei

```
lookAndFeel.frc
```

eingeführt. Eine Beschreibung ist im Abschnitt [Anpassung von Schriftarten](#) auf Seite 67 zu finden.

In den mitgelieferten Schriften mit den Dateinamen `FW-[M|L|H]-L2.fnt` wurde ein Fehler behoben, der sich in der Version 3.0.0 nicht ausgewirkt hat. Statt der Angabe von 0 (=ANSI_CHARSET) für das Attribute Charset ist in diesen Dateien der Wert 238 (EASTEUROPE_CHARSET) einzutragen. Bei den mit 3.1.0 ausgelieferten Dateien ist der Fehler bereits behoben. Wenn jedoch eigene Schriftarten auf Basis der Dateien aus der Version 3.0.0 erstellt wurden, dann ist die Fehlerbehebung in diesen Dateien nachzuziehen.

Im Rahmen der Umstellung auf andere Schriftarten wurde die Definition der Werte

```
TooltipFont,TooltipFontSize_M, TooltipFontSize_L, TooltipFontSize_H
```

in die Datei `lookAndFeel.frc` verlagert. Die Einstellungen in der anwendungsabhängigen Ressourcedatei entfallen damit. Durch diese Änderung kann die Schriftart von Tooltips pro LookAndFeel bestimmt werden.

7 Format der gepackten Bitmaps

Das Format der gepackten Bitmaps hat sich geändert. Deshalb müssen mit dem Einsatz der Version 3.1.0 alle Bitmaps neu gepackt werden. Der Grund dafür liegt darin, dass die Informationen zu den gepackten Bitmaps jetzt im Unicode-Zeichensatz abgelegt werden.

7.1 Neue Bitmaps

Die Bitmap mit dem Namen

001-0.bmp

ist ab Version 3.1.0 neu und bedingt deshalb ein erneutes Packen der Bitmaps. Sie findet Verwendung im led und dient zum Zeichnen des Hintergrunds im WYSISWG-Modus. Um einen anderen Hintergrund zu verwenden, kann die Bitmap ins Gruppenverzeichnis kopiert werden und angepasst werden. In Anwendungen kann dieses Grafikzeichen nicht genutzt werden.

8 Neue Einträge für Felder der Farbzordnungstabelle

Folgende Einträge sind neu bzw. haben sich in der Bedeutung geändert:

FIELD_LINE2
FIELD_LINE1_INVERS
FIELD_LINE2_INVERS

FIELD_LINE2 wird jetzt nicht mehr für das aktive (inverse) Feld verwendet, sondern für eine zweite Feldlinie an den Positionen $y+CellLineOffset-1$ und $y+CellHeight-CellLineOffset+1$.

Für das aktive Feld werden statt dessen die Farben der Einträge FIELD_LINE1_INVERS (alte innere Linie) und FIELD_LINE2_INVERS (neue äußere Linie) verwendet. Damit ist es möglich, für das aktive Feld eine dicke Linie (FIELD_LINE1_INVERS=BLACK, FIELD_LINE2_INVERS=BLACK) und für alle anderen Felder eine dünne Linie (FIELD_LINE1=BLACK, FIELD_LINE2=WHITE) zu definieren.

9 Belegung der Taste ESC

Durch den Eintrag

ESCAPE EN -- -- --

in der Datei `stdkey.fix` kann die Taste ESC mit FIX-Events (in diesem Beispiel EN) belegt werden.

10 Änderungen an Windows-Ressourcen

Die Ressourcen mit den IDs

FIXWINMENU/IDM_PROTOMSG (Menü)
IDD_PROTOMSG (Dialog)

sind neu. Wenn eine eigene DLL mit Ressourcen erstellt wurde, müssen sie in dieser ergänzt werden.

11 Anpassen der Benutzerbibliothek

Durch die Umstellung auf den Datentyp `wchar_t` hat sich die Schnittstelle zur Benutzerbibliothek geändert. Dabei wurde zur Umstellung das Makro

`_TCHAR`

verwendet. Dieses Makro wird entweder durch `char` oder durch `wchar_t` ersetzt je nachdem, wie das Makro `UNICODE` gesetzt wird. Dies ist das von Microsoft vorgegebene Verfahren zur Umstellung einer Anwendung auf Unicode. Durch das Umsetzen des Makros `UNICODE` ist es möglich, von der Anwendung eine Unicode- und eine Nicht-Unicode-Version zu erstellen. Die zweite Möglichkeit wird von FIX/Win 3.1.0 jedoch nicht genutzt. FIX/Win wird ab der Version 3.1.0 nur als Unicode-Version ausgeliefert.

Da Microsoft zusätzlich zu dem Makro `_TCHAR` auch die entsprechenden Makros für Funktionen anbietet, ist es leicht, den bestehenden Code auf Unicode umzusetzen.

Zur Umsetzung kann nach folgendem Schema vorgegangen werden:

Umsetzen der Funktionsdefinitionen

Nahezu alle Vorkommen von `char` in den Funktionsdefinitionen sind nach `_TCHAR` zu ändern, so dass diese zu den Prototypen passen, die in `include/fwown.h` und `include/fwownStd.h` definiert sind.

Ausnahmen von diesen Umsetzungen sind Funktionen, die als Parameter ein Bildschirmattribut bekommen. Hier wurde der Datentyp `unsigned char` in `unsigned short` geändert. Der Typ `wchar_t` entspricht zwar auch `unsigned short`, jedoch ist ein Attribut kein Zeichen, weshalb diese Abgrenzung vorgenommen wurde.

Die zweite Ausnahme ist die Funktion `FwownProcessData`. Da es sich hier um binäre Daten handelt, wurde der Datentyp für den Parameter `data (char**)` beibehalten.

Ersetzen von Funktionsaufrufen

Grundsätzlich bietet Microsoft für jede Funktionalität zwei Funktionen an:

- Eine, die Unicode nicht verwendet und den Datentyp `char` für die Parameter verwendet.
- Eine, die Unicode verwendet und den Datentyp `wchar_t` für die Parameter verwendet.

Zu diesen Funktionen gibt es ein Makro, das je nach Wert des Makros `UNICODE` zu einem der Funktionsaufrufe expandiert. Hinsichtlich der Namensgebung ist Microsoft jedoch zwei verschiedene Wege gegangen.

Bei dem ersten, der hauptsächlich für das WIN32-API verwendet wird, wird der Name des Makros aus dem ursprünglichen Namen gebildet. Der Name der Funktion, die Unicode nicht unterstützt, hat den Suffix `A`. Die Funktion die Unicode unterstützt, hat den Suffix `W`.

Beispiel

`DrawText` ist keine Funktion, sondern ein Makro, das entweder zu der Funktion

```
int DrawTextA(HDC hDC, LPCSTR lpString, int nCount, ...)
```

oder zu der Funktion

```
int DrawTextW(HDC hDC, LPCWSTR lpString, int nCount, ...)
```

expandiert.

Da auch in der Nicht-Unicode-Version schon das Makro verwendet wurde, ist zur Umstellung dieser Funktionen nichts weiter zu tun, als das Makro `UNICODE` zu setzen und dafür zu sorgen, dass die Parameter nicht von Typ `char` sondern vom Typ `_TCHAR` sind.

Bei den Funktionen der C-Laufzeitbibliothek ist Microsoft hinsichtlich der Namensgebung einen anderen Weg gegangen. Hier wird der ursprüngliche Name immer für die Funktion verwendet, die Unicode nicht unterstützt. Die Funktion, die Unicode unterstützt, bekommt `w` als Präfix im Namen oder `wcs` statt `str`. Für das Makro wird ein neuer Name verwendet, der je nach Funktion mit `_t` oder `_tcs` beginnt oder das `t` auch als zweiten Buchstaben verwendet. Die genauen Namen sind in der Microsoft-Dokumentation nachzulesen.

Beispiel

Statt `strcmp` ist das Makro `_tcscmp` zu verwenden. Es expandiert zu

```
wcscmp (const wchar_t *string1, const wchar_t *string2)
```

wenn das Makro `UNICODE` gesetzt ist und zu

```
strcmp (const char *string1, const char *string2)
```

wenn `UNICODE` nicht gesetzt ist.

Das bedeutet, dass im Code der Benutzerbibliothek sämtliche Aufrufe von C-Laufzeitfunktionen durch die entsprechenden Makros ersetzt werden müssen.

Ändern von Variablentypen

Zusätzlich zu den oben beschriebenen Schritten müssen nahezu alle Variablen des Typs `char` auf den Typ `_TCHAR` geändert werden. Ob für eine Variable diese Änderung notwendig ist oder nicht, kann nicht über eine einfache Regel bestimmt werden. Es ist zu untersuchen, wofür die Variable verwendet wird, wie sie beschrieben wird oder wo sie als Parameter für eine Funktion verwendet wird.

Anpassen des Zeichensatzes

Grundsätzlich verwendet `FIX/Win` für alle Zeichen den Datentyp `_TCHAR` und damit die Unicode-kodierte 16-Bit Werte. Wird innerhalb der Benutzerbibliothek ein anderer Zeichensatz benötigt (zB. um eine Logdatei in ANSI zu schreiben), dann ist eine Konvertierung vorzunehmen. Dazu eignen sich die Funktionen

```
MultiByteToWideChar
```

```
WideCharToMultiByte
```

des Windows-APIs sehr gut. Eine Beschreibung ist in der Microsoft-Dokumentation zu finden.

12 Erstellen von Protokoll-Profilen

Mit der Version 3.1.0 ist es möglich, ein Profil von dem Protokoll zwischen `FIX` und `FIX/Win` zu erstellen. Diese Möglichkeit wird im Abschnitt [Erstellen von Protokoll-Profilen](#) auf Seite 108 genauer beschrieben.

13 Starten eines Editors

Durch Aufruf der FIX-Funktion `editor()` wird ein externer Editor über FIX/Win gestartet. Im Abschnitt [Starten eines Editors](#) auf Seite 87 finden sich nähere Informationen dazu.

Anhang H Index

A	
Anpassung von FIX/Win	41
Applikationsserver	10
Arbeitsverzeichnis	14
Ausgabelisten	39
B	
Beispiel	
eigene Semigrafikzeichen	58
neue Tastaturbelegung	56
Benutzerverzeichnis	81
Bereichsmarkierungsmodus	34
Bildgröße	10, 36, 38
Bildstil	10, 37, 38
Bitmappeditor	75
Bitmappeditor für Entwicklermenü	75
Bitmaps	
für Icons	50
für Semigrafikzeichen	57
für Tasten-Label	54
über Entwicklermenü editieren	75
über Entwicklermenü nachladen	76
bmpack	60
BmPackOptions Ressource	62
C	
CallBack	90
Client-Bereich	11
Container-Fenster	11
D	
Datei	
Bitmap für Icon	47
Bitmaps für Semigrafikzeichen	57
Bitmaps für Tasten-Label	54
Tastenbelegung	53
Display-Host	10
E	
Einstellungen	38
enable mouse	51
Entwicklermenü	75
F	
Farbzuordnungstabelle	63
Featurelevel	12
Feldmarkierungsmodus	34
Festplattenspeicher	9
FIX Events	
Tabelle	53
FIX/Win-Fenster	11
FIX-Iconset	45
Frontend	25
Frontend Host	10
Funktionsaufrufe, eigene	20
FW_GETCLIENTHWND	102
FW_GETCOLOR	100
FW_GETCONTAINERHWND	102
FW_GETFIXHWND	99
FW_GETFRAMERESINST	98
FW_GETGROUP	99
FW_GETHOSTNAME	99
FW_GETICONHWND	99
FW_GETMAINFRAMEHWND	102
FW_GETMSGHWND	102
FW_GETPRGTAB	101
FW_GETRESINSTANCE	99
FW_GETSCREENINFO	100
FW_GETUSER	98
FW_GETUSERDIR	101
FW_MSGWINACTIVATE	102
FW_MSGWINSHOWMSG	103
FW_RESIZEBUFFER	101
FW_SENDEVENT	101
FW_SHOWMESSAGEBOX	104
FW_STATBOXSETMSG	103
FWBMPTOOL	75
FWEDITOOL	75
FWFKT_GETCLIENTHWND	102
FWFKT_GETCOLOR	100
FWFKT_GETCONTAINERHWND	102
FWFKT_GETFIXHWND	99
FWFKT_GETFRAMERESINST	98
FWFKT_GETGROUP	99
FWFKT_GETHOSTNAME	99
FWFKT_GETICONHWND	99
FWFKT_GETMAINFRAMEHWND	102
FWFKT_GETMSGHWND	102
FWFKT_GETPRGTAB	101
FWFKT_GETRESINSTANCE	99
FWFKT_GETSCREENINFO	100
FWFKT_GETUSER	98
FWFKT_GETUSERDIR	101
FWFKT_MSGWINACTIVATE	102
FWFKT_MSGWINSHOWMSG	103
FWFKT_RESIZEBUFFER	101
FWFKT_SENDEVENT	101
FWFKT_SHOWMESSAGEBOX	104
FWFKT_STATBOXSETMSG	103
fown.h	89
FOWN_EXPORTS	89
FwownAddPaintArea	95
FwownDelPaintArea	96, 103
FwownDrawPaintArea	92
FwownExec	91
FwownGetVersionInfo	91
FwownHook	96
FwownProcessData	91
fownStd.h	89
FwownUpdatePaintArea	95
fx_exec_frontend()	91
fx_transfer_to_frontend	92
FXTERMPATH	25
G	
GetCharSet	100
GetClientHwnd	102
GetColor	100
GetContainerHwnd	102
GetFixHwnd	99
GetFrameResInst	98
getFrontendData	92
GetGroup	99
GetHostname	98
GetIconHwnd	99
GetMainFrameHwnd	102
GetMsgWinHwnd	102
GetPassword	98
GetPrgtab	101
GetResInstance	99
GetScreenInfo	100
GetUserDir	101
GetUsername	98
Gruppenverzeichnis	14, 23, 41
H	
Hauptfenster	11
Hilfe	38
Hinweistext zu Icons	47

Hostname	23	Lesen	80
I		Schreiben	81
Icon		Ressourcen	
Hinweistext	47	anwendungsabhängig	82
IconButton	11	anwendungsunabhängig	82
Iconbox	33, 38	sonstige	83
Änderungen	48	vordefiniert	83
Iconboxinfo	77	Ressourcenname	79
Icondefinitionsdatei	46	Ressourcewert	79
Iconliste	45	Ressourcewerte	
Iconset	45, 51	Zugriff auf andere	79
von FIX vorbelegt	45	Ressourcezuweisung	79
K		S	
K_SB (Event)	92	Schalter -service	25
Kommandozeile	19	Semigrafikzeichen	
Kommandozeilenparameter		Größen der Bitmaps	58
/fwown	89	über Entwicklermenü editieren	75
Kommentarzeilen	23	Verwendung von eigenen	57
Konfiguration		SendEvent	100
über Entwicklermenü bearbeiten	76	service (Schalter)	25
Kopiermodus		SetStatboxMsg	103
Einschalten über Timeout	37	ShowJumpCursor	52
L		ShowMessageBox	104
LOGNAME	19, 27, 31	ShowMsg	103
Logo, Erstellen mit Semigrafikzeichen	58	SPACE	47
M		Standardkonfiguration	14
Mausbedienung	29	Startmeldungen	25
maussensitiven Masken	51	Startscript	23, 24
Meldungsfenster	11, 32	Statusfenster	38
Menüaufbau	31	Statuszeile	11
Menüpunkt		stdkey.fix	56
Anzeige	32	T	
Bild neu aufbauen	33	Tastaturbedienung	28
Bildgröße	36	Tastaturbelegung, Erstellen von eigener	53
Bitmaps editieren	75	Tastenbelegung	10
Einstellungen	36	Tasten-Label	54
FIX-WIN	31	Größe der Bitmaps	55
Hilfe	38	Textausgabe des FIX-Programms	38
Hilfe für Entwickler	77	Texteditor für Entwicklermenü	75
Iconbox	33	U	
Konfiguration editieren - Nachladen	76	Umgebungsvariable	
Meldungsfenster	32	FWBMPTOOL	75
Programmtabelle editieren	75	FWEDITOOL	75
Statusfenster	33	FXTERMPATH	25
Stil	37	LOGNAME	19, 27, 31
Übersicht	38	USER	19, 27, 31
Verbindung abbrechen	31	USER	19, 27, 31
Verbindung aufbauen	31	User-Iconset	45
Verlassen	31	V	
Modifier Tasten Kürzel	55	Verbindungsaufbau	27, 31
MsgBox	25	Version	12
MsgWinActivate	102	Verzeichnis	
O		für Bitmaps der Semigrafikzeichen	57
Objektmarkierungsmodus	34	W	
P		Windows-Tasten Kürzel	53
p_callID	90	Z	
Packen von Bitmaps	60	Zeicheninfo	77
Passwort	27		
Patchdate	12		
Port-ID	12		
R			
Rahmenmarkierungsmodus	36		
ResizeBuffer	92, 101		
Ressourcedatei			
Arten von	80		
Aufbau	79		
Einlesereihenfolge	82		
Einschließen von	81		
fw_prg.frc	81		
fw_usr.frc	80		
include	80		