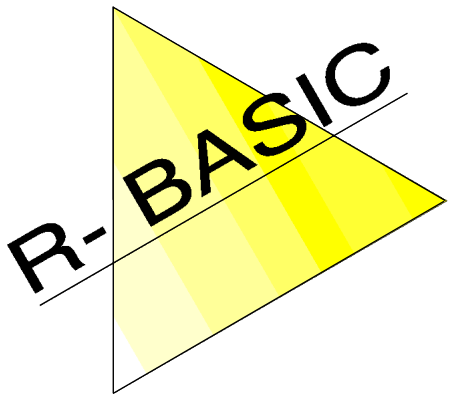
R-BASIC

Einfach unter PC/GEOS programmieren



Benutzer-Handbuch

Volume 1
Einführung, Die IDE, Eigenständige Programme

Version 1.0

(Leerseite)

Inhaltsverzeichnis

(Leerseite)

1 Einführung in R-BASIC

1.1 Was ist R-BASIC

R-BASIC ist eine neue Programmiersprache für PC/GEOS. Sie ist einfach zu lernen und einfach zu benutzen, so dass Anfänger gut mit ihr zurechtkommen. Gleichzeitig ist sie so leistungsfähig, dass auch anspruchsvolle Projekte realisierbar sind. R-BASIC erfordert mindestens eine PC/GEOS Version ab GeoWorks 2.01 (oder höher). Empfohlen wird eine der aktuellen BreadBox Ensemble Versionen.

Warum gerade ein BASIC?

1. Einfache Syntax. Die BASIC Syntax ist leicht zu erlernen, der Funktionsumfang von R-BASIC ermöglicht trotzdem die Programmierung von umfangreichen und professionellen Projekten.
2. Kompatibilität. Es gibt eine Menge "alter", aber trotzdem interessanter Programme, die nur darauf warten, nach R-BASIC portiert zu werden.
3. Man kann quasi jedes beliebige BASIC Handbuch hernehmen, um die Grundlagen der (R-)BASIC Programmierung zu erlernen. Die Details mögen verschieden sein, aber die Konzepte, wie man programmiert, bleiben gleich.

Ist R-BASIC objektorientiert?

Ja, Sie haben Zugriff auf das leistungsfähige PC/GEOS Objekt-System. Die in R-BASIC verfügbaren Objektklassen werden im Objekthandbuch beschrieben.

Was sollte der Einsteiger wissen?

- Konzept: Nur eine Quellcode-Datei
  Bei den meisten R-BASIC Projekten befindet sich der komplette Quellcode und auch die zugehörigen UI-Objekte (UI: User-Interface) in einer einzigen Datei. Sie können hier sogar kleinere Grafiken ablegen (Stichwort: PictureList). Natürlich können Sie bei größeren Projekten auch auf externe Dateien und auf Libraries zurückgreifen.

- Konzept: Einfache Struktur
  Durch die Verwendung einer BASIC-Syntax sind die Programme gut lesbar. Auf die Unterscheidung von Groß- und Kleinschreibung wird verzichtet. Das kommt besonders Anfängern sehr entgegen.
  In Zahlen wird als Dezimaltrennzeichen immer der Punkt (niemals das Komma!) verwendet. Damit sind R-BASIC Programme auf alle PC/GOS Systeme übertragbar, egal welche System-Einstellungen der Nutzer hat.

- Konzept: Action-Handler
  Jedes Mal, wenn der Nutzer z.B. einen Button anklickt, wird eine R-BASIC Routine gerufen, welche die zugehörige Funktion ausführt. Diese Routinen werden "Action-Handler" genannt, weil sie die Aktion (Anklicken des Buttons) behandeln. Im Kern besteht jedes R-BASIC Programm aus einer Menge von Action-Handlern, die zum gegebenen Zeitpunkt ausgeführt werden.

- Konzept: Eigenständige Programme
  Wenn Ihr Programm fertig ist können Sie daraus ein "eigenständiges Programm" erzeugen. Diese "R-Apps" sind auf jedem PC/GEOS-System, auch ohne die R-BASIC Entwicklungsumgebung, lauffähig. Aus der Sicht des Nutzers unterscheiden sich R-App's nicht von "normalen" Programmen, die mit dem PC/GEOS SDK geschrieben wurden. Die R-BASIC IDE unterstützt Sie dabei, indem sie ein Installer-Paket für den Universal-Installer anlegt, dass automatisch alle benötigten Dateien enthält.

- Konzept: Systemintegration
  Neben den für eine Programmiersprache unverzichtbaren Elementen (Schleifen, Unterprogramme, Mathematik, Stringfunktionen, Dateiarbeit usw.) bietet R-BASIC viele Funktionen an, die das PC/GEOS System zur Verfügung stellt. Dazu gehören der Zugriff auf die Zwischenablage, die Verwendung von Hilfedateien, der Zugriff auf den Drucker, das Setzen von Timern und die Möglichkeit, in das Handling von Maus- und Tastaturereignissen einzugreifen. Und natürlich können Sie Objekte verwenden, so das R-BASIC Programme genau so aussehen, wie mit dem PC/GEOS SDK geschriebene Programme.

- Konzept: Einfaches Übersetzten
  Das Programm "R-BASIC Translator" erlaubt es, fertige R-Apps in eine andere Sprache zu übersetzten. Sie müssen dazu im Programmcode keinerlei Vorkehrungen treffen. Der "R-BASIC Translator" muss separat von der R-BASIC Webseite heruntergeladen werden. Eine Beschreibung, wie man beim Übersetzen von R-BASIC Programmen vorgehen muss, finden Sie im Kapitel 4 dieses Handbuchs.

Wie bekomme ich Hilfe?

Für den Einsteiger ist unbedingt das Kapitel 1.2 (Womit beginnen) zu empfehlen. Später werden Sie das Hilfesystem von R-BASIC zu schätzen lernen. Es besteht aus drei Elementen:

- Die "Wizzards" enthalten die Syntax und eine Kurzbeschreibung alle R-BASIC Befehle, Objektklassen, Instancevariablen und so weiter. Sie sind über das Hilfe-Menü oder die Taste F2 erreichbar.
- Die Handbücher befinden sich im Ordner R-BASIC\Dokumente. Hier werden alle Details der R-BASIC Sprache und des R-BASIC Objekt-Systems ausführlich beschrieben.
- Die Hilfe-Datei ist über die Taste F1 oder über das Hilfe-Menü erreichbar. Hier liegt der Schwerpunkt auf der thematischen Sortierung der Informationen.

Zusätzlich sind eine Menge an Beispielen zu den verschiedenen Themen im Ordner R-BASIC\Beispiel verfügbar.

Wie erstellt man ein R-BASIC Programm?

- Zu jedem BASIC-Programm gehören zwei Teile: der eigentliche Programmcode und die UI-Objekte. Die UI-Objekte werden im Fenster "UI-Objekte" vereinbart. Der Programmcode selbst wird in bis zu 5 weiteren Fenstern geschrieben.

- Nachdem Sie ihren Code geschrieben und die UI-Objekte deklariert haben müssen Sie das Programm "compilieren". Dabei wird die Syntax geprüft und eine Sequenz von "Tokens" erzeugt, die später "interpretiert" (ausgeführt) werden können. Dieses Konzept wird zum Beispiel auch von der Programmiersprache Java eingesetzt.

- Nachdem das Compilieren erfolgreich war können Sie Ihr Programm starten. Sie können jetzt prüfen, ob alle Programmfunktionen so arbeiten, wie Sie es vorgesehen haben.

- Wenn alles funktioniert können Sie eine R-App anlegen. R-Apps bestehen aus drei wesentlichen Teilen: dem Launcher, der BIN-Datei und den erforderlichen Libraries.
  Der Launcher ist der für den Nutzer sichtbare Teil. Er befindet sich im World-Ordner. In der BIN-Datei (binäre Daten-Datei) befindet sich das übersetzte BASIC-Programm. Der Launcher lädt diese Datei und führt den darin enthalten Code aus. Die genauen Zusammenhänge sind im Kapitel 3 dieses Handbuchs erklärt.

1.2 Womit beginnen?

Die BASIC-Handbücher sind sehr umfangreich. Für einen Einsteiger ist es sehr schwer zu entscheiden, was er zuerst lesen soll und was sich eher für fortgeschrittene Programmierer eignet. In diesem Kapitel finden Sie eine Liste der für den Einstieg empfohlenen Themen.

Orientierung in den Handbüchern

Neben den Handbüchern selbst gibt es mehrere Möglichkeiten, sich einen Überblick über die vorhandenen Funktionen zu einem Thema zu verschaffen oder die passenden Kapitel im Handbuch zu finden.

- Die **Wizzards** (Menü: Hilfe) sind der Kern des BASIC Hilfe-Systems. Sie sind thematisch (BASIC Wizzard) bzw. alphabetisch nach Objektklassen (Objekt Wizzard) sortiert. Dadurch sind sie hervorragend geeignet, sich einen Überblick über die vorhandenen Funktionen zu einem Thema oder zu den Fähigkeiten und Eigenschaften einer Objektklasse zu verschaffen.

- Die **Hilfe-Datei** erfüllt drei Aufgaben:
  1. Sie ist als Wegweiser durch die Manuals konzipiert. Hier finden Sie - thematisch sortiert - die wesentlichen zu einem Thema gehörenden Handbuchkapitel aufgelistet. Zum Beispiel wird das Thema Grafik sowohl im Programmierhandbuch (grundlegende Befehle) als auch im Objekt-Handbuch (Objekte zur Grafikausgabe) und im Handbuch Spezielle Themen (Block-Grafik) behandelt.
  2. Sie enthält eine thematisch sortierte Liste alle R-BASIC Befehle und ihrer Syntax.
  3. Sie enthält eine Übersicht über alle R-BASIC Objektklassen, ihrer Instancevariablen und der dazugehörigen Konstanten.

- Die Dokumente **R-BASIC Kommando Übersicht** und **R-BASIC Objekt Übersicht** sind zu empfehlen, wenn Sie bereits etwas Erfahrung mit R-BASIC haben und sich einen Überblick verschaffen wollen oder wenn Sie ein spezielles Problem haben und nicht so genau wissen, wo sie nach einer Lösung suchen sollen. Suchen Sie nach Befehlen oder Instancevariablen, die so klingen, als könnten sie helfen und informieren Sie sich dann im Handbuch über die Details.

Tutorials

Tutorials enthalten Einführungen in die Programmierung mit R-BASIC am Beispiel. Wenn sie keine Programmiererfahrungen haben sollten Sie diese Kapitel lesen, auch wenn Sie vielleicht nicht jeden einzelnen Schritt verstehen. Auch Leser mit Programmiererfahrung, die neu in der R-BASIC-Programmierung sind, sollten die Tutorials lesen.

Programmierhandbuch: **Kapitel 1: Ein Anfängertutorial**
Objekthandbuch: **Kapitel 1.1: Ein Beispiel zur Einführung**

Handbücher

Die folgenden Kapitel enthalten grundlegende Informationen, die man für die meisten Programmierprojekte benötigt. Sie sollten diese Kapitel zumindest überfliegen. Auch hier gilt: Sie müssen nicht alles am Anfang verstehen und Sie müssen auch nicht alles gelesen haben, bevor Sie mit dem Programmieren anfangen.

Programmierhandbuch: Konzeptionelles zur Basic Programmiersprache

Kapitel 2.1: **Grundlegende Konzepte**
Kapitel 2.2 **Variablen und Typen**
- Empfohlen für den Einstieg:
  - 2.2.1 **Was sind Variablen?**
  - 2.2.2 **Numerische Datentypen und numerische Ausdrücke**
  - 2.2.3 **Stringtypen und Stringausdrücke**
  - 2.2.10 **Die CONST Anweisung**

Im weiteren Verlauf sind die folgenden Kapitel zu empfehlen:
- 2.3 **Arbeit mit numerischen Ausdrücken**
- 2.4 **Arbeit mit Strings**
- 2.5 **Programmablaufsteuerung**
- 2.6 **Unterprogramme**

Objekthandbuch: Konzeptionelles zu Objekten

Kapitel 1: **Überblick** (komplett)
Kapitel 2.1: **Objekte und Objekt-Bäume**
- Empfohlen für den Einstieg:
  - 2.1.1 **Überblick**
  - 2.1.2 **Arbeit mit Objekten**
- Die anderen Unterkapitel sind nicht für Anfänger geeignet!

Kapitel 2.2 **Ausgabe von Grafik**
Kapitel 3.1 **Caption: Die Objekt-Beschriftung**
Kapitel 3.2 **Objekt States**
Kapitel 3.3 **Geometriemanagement**
- Empfohlen für den Einstieg:
  - 3.3.1 **Überblick**
  - 3.3.2 **Angabe von Größen, Positionen und Abständen**
  - 3.3.3 **Anordnung der Objekte** (zumindest 3.3.3.1 und 3.3.3.2)

Objekthandbuch: Objekte

Beschäftigen Sie sich mit einem speziellen Objekt dann näher, wenn Sie es einsetzen wollen. Für den Anfang benötigen Sie wahrscheinlich die Objekte **Application**, **Primary**, **Button**, **Group**, **Menu**, **View** und **BitmapContent**.
Wichtig: GEOS Objekte sind in ihrer Grundfunktion einfach einzusetzen, können aber auch sehr komplexe Seiten haben. Für den Anfang müssen Sie nicht alles zu einem Objekt lesen oder gar verstehen.

Sonstiges

Es ist zu empfehlen, ein Kapitel dann gründlich zu lesen, wenn Sie sich **erstmalig mit der Thematik** beschäftigen. Die meisten Kapitel gehen vom Allgemeinen zum Konkreten und vom Leichteren zum Komplizierten, so dass der Einsteiger zumindest die ersten Abschnitte gut verstehen kann.

Auch wenn Sie einen **Befehl** einsetzen wollen, den Sie noch nie benutzt haben, sollten Sie zuerst das Handbuch konsultieren. In den meisten Fällen finden Sie hier einfache Beispiele. Manchmal gibt es auch Syntaxvarianten oder Eigenschaften, die nicht auf den ersten Blick offensichtlich sind.

Nutzen Sie die **Beispiele**! Es ist erlaubt und gewünscht, den Beispielcode in eigene Projekte einzubinden, ihn dabei zu modifizieren und ihn an andere Programmierer weiterzugeben.

1.3 Wichtige Pfade und Ordner

Dieses Kapitel listet die Pfade und Ordner auf, mit denen R-BASIC arbeitet. Sie dürfen die hier aufgeführten Ordner weder umbenennen noch verschieben, sonst kann R-BASIC nicht arbeiten.

Grundsätzlich gilt, dass jede neue R-BASIC Version alle mitgelieferten Dateien überschreiben kann. Sie müssen also geänderte Dateien (Beispiele, Dokumente, Fonts usw.) entweder umbenennen oder an einem anderen Ort speichern.

Document\R-BASIC

In diesem Ordner und seinen Unterordnern finden Sie alle R-BASIC Dokumente und Programme. Per Default legt R-BASIC hier auch neue, leere Dokumente ab. Diese Einstellung können Sie im Menü "Optionen" - "Programm Optionen" - "Default Programm Pfade" ändern.
Für Ihre eigenen Projekte sollten Sie hier einen eigenen Unterordner anlegen.

Document\R-BASIC\Beispiel

Hier finden Sie die von R-BASIC mitgelieferten Beispiele. Sie dürfen hier weitere Unterordner mit eigenen oder geänderten Beispielen anlegen.

Document\R-BASIC\Dokumentation

Dieser Ordner enthält die Handbücher und alle mit R-BASIC mitgelieferten Dokumente. Zusatzprogramme wie Libraries können hier ihre eigenen Dokumente ablegen. Auch Sie können hier zusätzliche, eigene Dateien ablegen.

USERDATA\R-BASIC

Dieser Ordner und seine Unterordner enthalten systemwichtige Dateien der R-BASIC IDE sowie die compilierten Programme. Es ist nur in wenigen Fällen erforderlich, dass Sie hier manuell Änderungen vornehmen.

USERDATA\R-BASIC\BIN

Hier finden Sie die BIN-Dateien der compilierten, eigenständigen Programme. Es wird empfohlen, dass jeder Programmierer einen eigenen Unterordner für alle seine Programme verwendet. Das kann beim Erzeugen der eigenständigen Programme (siehe Kapitel 3) eingestellt werden, das heißt, Sie müssen den Unterordner nicht manuell anlegen.
Beispiel:
Der Ordner **USERDATA\R-BASIC\BIN\MAXMU** enthält alle BIN-Dateien der Programme von Max Mustermann. Nur für das Programm SuperGame, das neben der BIN-Datei noch viele Level-Dateien enthält, hat er einen eigenen Unterorder angelegt: **USERDATA\R-BASIC\MAXMU\SuperGame**.

USERDATA\R-BASIC\FONT

Dieser Ordner enthält die Blockgrafik-Fonts (RBF-Dateien). Sie sollten alle eigenen Blockgrafik-Fonts hier ablegen.

USERDATA\R-BASIC\IMAGES

Hier legt der R-BASIC-Installer Grafiken ab, die für alle BASIC-Programmierer verfügbar sein sollen. Sie können hier auch ihre eigen Grafiksammlung, z.B. einen Ordner mit *.ICO-Dateien, anlegen. Zum Zeichnen der Grafiken können Sie z.B. den R-BASIC Befehl DrawImage oder die Objekt-Instancevariable CaptionImage verwenden.

Beachten Sie, dass die Bilder nicht automatisch in das Installer-Paket der R-App aufgenommen werden. Sie können das beim Anlegen des Installer-Paktes (siehe Kapitel 3.2) sehr einfach angeben.

Die R-BASIC Standard-Installation legt folgenden Unterordner an:

IMAGES\Icon Tool Graphics: Dieser Ordner enthält die Bilder der Werkzeuge aus der Werkzeugleiste (i.a. 15x15 Pixel groß). R-BASIC hat für die Werkzeuge keine eigenen Objekte. Sie können die Bilder aus diesem Ordner als Captions für eigene Buttons verwenden, um ihrem Programm systemkonform aussehen zu lassen.

In der Standard-Installation sind folgende Unterordner enthalten:

Icon Tool Graphics\CD: Bilder für Buttons, die für einen CD-Player oder ein ähnliches Programm benötigt werden (Start, Stopp, schneller Vorlauf ...).

Icon Tool Graphics\Document: Bilder für Buttons, die zur Dokumentenverwaltung (Neu, Öffnen, Speichern usw.) benötigt werden.

Icon Tool Graphics\Edit and more: Bilder für Bearbeiten-Buttons, z.B. Kopieren, Einfügen, Linienstil usw.

Das Zusatzpaket "More Tool Images", das von der R-BASIC Webseite heruntergeladen werden kann, legt die folgenden Unterordner an:

IMAGES\Cards:
Spielkarten Vorder- und Rückseiten

IMAGES\Icon Tool Graphics\Bitmap: Bilder für die Bitmap-Tools

IMAGES\Icon Tool Graphics\Colors: Bilder für die Farbauswahl-Tools

IMAGES\Icon Tool Graphics\Object Tools: Bilder für die Objekt-Tools wie Spiegeln, Zeiger-Werkzeug, Objekte verbinden usw.

IMAGES\Icon Tool Graphics\Text: Bilder für die Text-Tools wie fett, tiefgestellt, durchgestrichen usw.

IMAGES\Icon Tool Graphics\Other: Bilder für weitere Tools wie Lupe, Rechtschreibprüfung, Fenster überlappen usw.

USERDATA\R-BASIC\Library\BIN

Die BIN-Dateien von Libraries werden automatisch hier abgelegt. Hier müssen auch die für R-BASIC geschriebenen SDK-Libraries abgelegt werden.

USERDATA\R-BASIC\Library\System

Der Ordner enthält die Quellcodes der R-BASIC Systemlibraries. Sie können die Quellcodes einsehen, dürfen aber keine Änderungen vornehmen!

USERDATA\R-BASIC\RunTemp

Hier legen alle R-BASIC-Programme ihre temporären Dateien ab. Damit der Ordner nicht überquillt wird er bei Gelegenheit automatisch aufgeräumt.

USERDATA\R-BASIC\Syntax

Hier liegen die Dateien mit den Farbschemata der Syntax-Hervorhebung. Im Menü "Optionen" - "Editor-Einstellungen" - "Farben" können Sie die Farben ändern, Farbschemata anlegen und laden.

USERDATA\R-BASIC\Tools

Hier werden Programme (oder Links auf Programme) abgelegt, die im Menü "Extras" - "Tools" erscheinen sollen. Sie können hier eigene Programme ablegen.

USERDATA\R-BASIC\Wizzard

Wenn Sie eine Library schreiben, müssen Sie den Wizzard zu dieser Library hier ablegen. Die R-BASIC IDE erkennt diesen Wizzard dann automatisch und liest ihn beim nächsten Start ein.

WIZZARD\Core: Dies ist ein Systemordner. Er enthält die Wizzard-Dateien des R-BASIC Kerns und der Objekte.

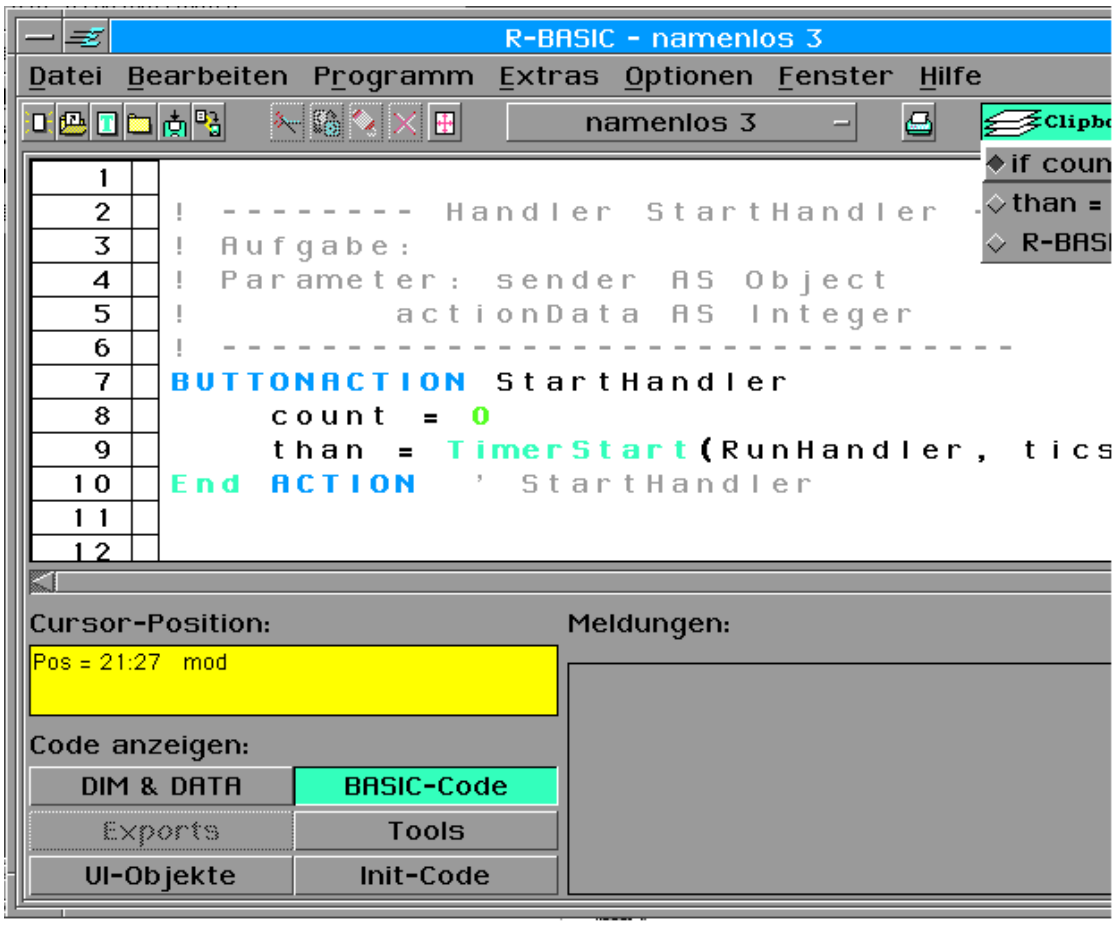
WIZZARD\SysLibs: Dies ist ein Systemordner. Er enthält die Wizzard-Dateien der R-BASIC System-Libraries.

(Leerseite)

2 Die R-BASIC IDE

2.1 Bedienung der IDE

Die IDE ist die integrierte Entwicklungsumgebung (Integrated Development Environment) von R-BASIC. Sie enthält alle Werkzeuge, die Sie zum Schreiben eines R-BASIC Programms benötigen.



Im Wesentlichen besteht die IDE aus vier Teilen:

- Ganz oben finden Sie die Menü-Leiste. Eine Beschreibung der einzelnen Menüpunkt finden Sie im nächsten Kapitel (2.2).
- Darunter liegt die Tool-Leiste. Neben den aus anderen Anwendungen bekannten Werkzeugen zum Verwalten von Dokumenten, zum Bearbeiten und zum Drucken finden Sie hier ein Werkzeug zum Überwachen der Zwischenablage. Dieses wird im Folgenden beschrieben.
- Den Hauptteil nimmt das Editor-Fenster ein. Hier definieren Sie die UI-Objekte und schreiben den Programmcode.
- Unten befindet sich der Status-Bereich. Er zeigt die Cursor-Position, ob und wieviel Text Sie selektiert haben und welches Code-Window aktuell aktiv ist. Außerdem werden hier alle Compilermeldungen ausgegeben.
  Eine Beschreibung der Code-Windows finden Sie etwas weiter unten.

Bedienung des Editors

Der R-BASIC Editor ist mehr als ein einfacher Text-Editor. Zu den wesentlichen Besonderheiten gehören:

• Mehrstufige Undokette

Die R-BASIC IDE verfügt über eine mehr als 100 Stufen tiefe Undokette. Sie können also mehr als 100 Eingabeoperationen, Einfügen, Löschen usw. zurücknehmen.

• Das Clipboard History Werkzeug

Jedes Mal, wenn Sie einen Text in die Zwischenablage kopieren, erzeugt R-BASIC einen Eintrag in die Liste des Clipboard-History Werkzeugs. Damit können Sie auf ältere Inhalte der Zwischenablage zugreifen, die das System längst vergessen hat. Um einen Eintrag aus der Liste in den Quelltext einzufügen müssen Sie ihn nur anklicken.

 Optionen für das Clipboard-History Werkzeug

Die Bezeichnungen sollten selbsterklärend sein.

 Einfügen-Schalter

Falls die Option "Bei Auswahl sofort Einfügen" nicht aktiv ist müssen Sie diesen Schalter anklicken.

• Tastaturkommandos

Neben den üblichen Tastaturbefehlen (Einfügen, Ausschneiden usw.) verfügt die R-BASIC IDE zum Beispiel über Befehle zum Ausrichten oder Verschieben von ganzen Textblöcken. Die wichtigsten finden Sie im Menü "Bearbeiten" (Tastaturbefehl Strg-K), eine vollständige Liste finden Sie im Hilfe-Menü.

• Zeilenverlängerung mit \

Jede Codezeile kann bis zu 256 Zeichen lang sein. Da das sehr unübersichtlich sein kann können Sie eine Codezeile auf mehrere Editorzeilen aufteilen, indem Sie am Ende der Zeilen das Fortsetzungszeichen \ eingeben. Der Compiler behandelt so verbundene Zeile wie eine einzige Zeile. Die Kombination

```
y =  5*sin(2*PI*t) + \
          3*cos(4*PI*t) + \
          sin(PI*t + PI/2)
```

behandelt der Compiler genau so als hätten Sie geschrieben:

```
y = 5*sin(2*PI*t) + 3*cos(4*PI*t) + sin(PI*t + PI/2)
```

Sie sollten auf diese Weise keine Stringkonstanten (z.B. den Text einer MsgBox-Anweisung) aufzuteilen. Besser ist es, mehrere Strings zu verwenden, die durch ein + kombiniert werden. Der Compiler setzt die Strings automatisch zu einem String zusammen. Ein Beispiel:

```
Memo MyMemo
 text$ = "Prinzipiell ist es möglich auf diese Weise auch " \
     + "Stringkonstanten "\
     + "aufzuteilen. Das ist jedoch schlechter Stil."
 End Object
```

Das Verbinden mehrerer Zeilen funktioniert sowohl im normalen BASIC-Code als auch im UI-Code. Es gibt jedoch eine Ausnahme: Im UI-Code dürfen UI-Anweisungen, die sich auf ein anders Objekt beziehen (z.B. Children, Content, ...), nicht Teil von auf diese Weise verbundenen Zeilen sein. Der Compiler quittiert das mit völlig unsinnigen Fehlermeldungen. Außerdem sollte man wissen, dass innerhalb von verbundenen Zeilen auftretende Compiler- oder Laufzeitfehler immer in der letzten der verbundenen Zeilen gemeldet werden.

• Codebausteine

Im Menü "Extras" finden Sie den Menüpunkt "Codebausteine". Hier können Sie typische Programmbausteine, Objekte und ganze Codesequenzen in Ihren Programmcode einfügen. Damit ersparen Sie sich sehr viel Arbeit.

• Kommentarzeilen

Zeilen oder Zeilenanschnitte, die mit dem Wort REM (engl. remark: Bemerkung) oder einem der Zeichen ! (Anführungszeichen) oder ’ (Hochkomma) beginnen, interpretiert der Compiler als Kommentar. Kommentare und Leerzeilen verbessern die Lesbarkeit eines Programms deutlich, werden vom Compiler aber ignoriert. Sie verlangsamen das Programm auch nicht.

Verwendung der Code-Fenster

Wenn Sie ein Programm schreiben stehen Ihnen bis zu 6 Code-Fenster zur Verfügung.

UI-Objekte

In diesem Fenster müssen die UI-Objekte des Programms vereinbart werden. Das Schreiben von Code in diesem Fenster ist nicht möglich.

DIM & DATA

Bei umfangreichen Projekten sollten hier global Deklarationen untergebracht werden. Das Schreiben von Code ist möglich, aber ganz schlechter Stil.

Exports

Wenn Sie eine Library schreiben werden hier Deklarationen untergebracht, die von der Library "exportiert" werden. Libraries sind im Kapitel 2.12 beschrieben.

BASIC-Code, Tools, Init-Code

Diese Fenster sind für den eigentlichen Programmcode vorgesehen. Aus Sicht von R-BASIC sind diese Codefenster alle gleichwertig. Einem erfahrenen Programmierer ermöglichen sie, seinen Code übersichtlicher zu gestalten. Programmieranfängern wird empfohlen zunächst nur das Fenster BASIC-Code zu benutzen.

Die Fenster werden in der Reihenfolge "Exports" -> "DIM & DATA" -> "UI-Objekte" -> "Tools" -> "BASIC-Code" -> "Init" compiliert. Dadurch stehen den drei eigentlichen Codefenstern sowohl die Vereinbarungen aus Exports und DIM & DATA als auch die Namen aller UI-Objekte zur Verfügung.

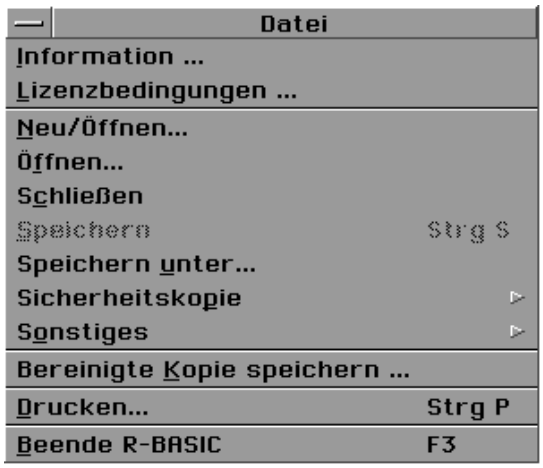
Tipps:

- Die Codefenster können auch mit den Tastenkombinationen Strg-1 bis Strg-6 angewählt werden.
- Im Menü "Optionen" -> "Editor-Einstellungen" finden Sie den Menüpunkt "Code-Windows umbenennen". Dort können Sie den Windows "DIM & DATA", "BASIC-Code", "Tools" und "Init" andere, an ihr aktuelles Projekt angepasste Namen geben.

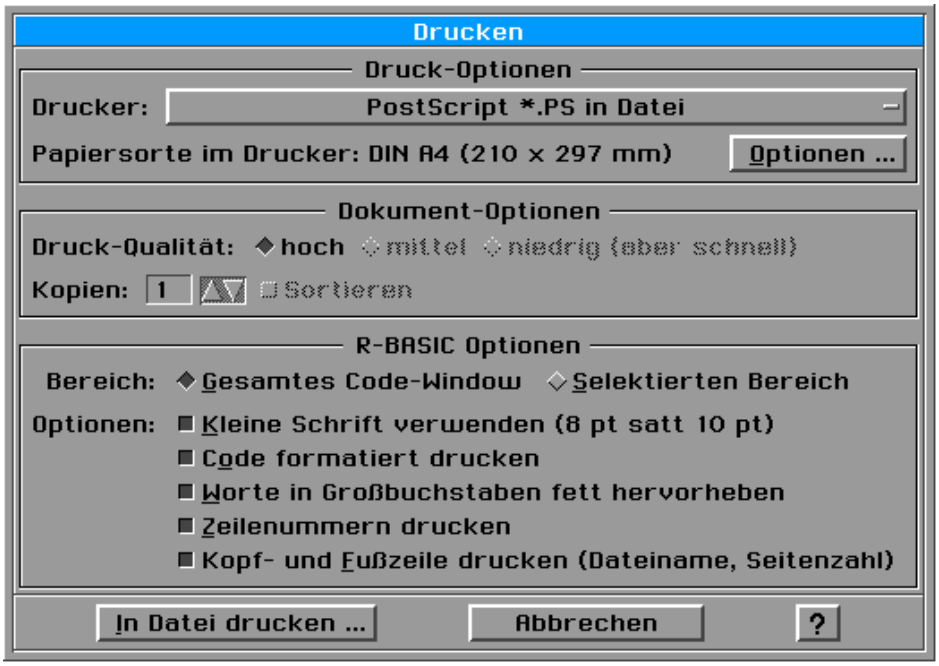
2.2 Die Menüs

Das Datei-Menü

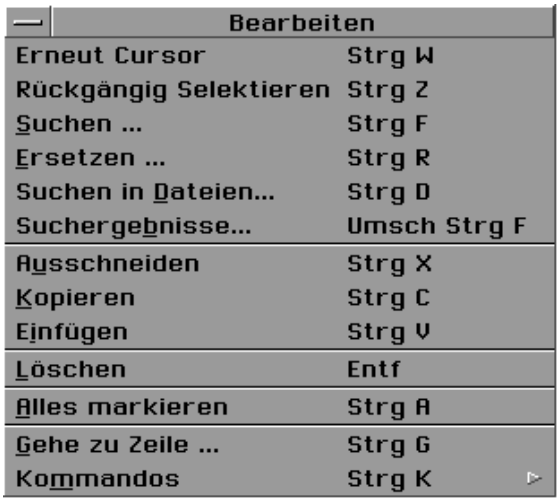
Im Datei-Menü finden Sie die üblichen Funktionen zum Verwalten der Dokumente des Programms. In unserem Fall sind das die Dateien mit dem Quellcode.



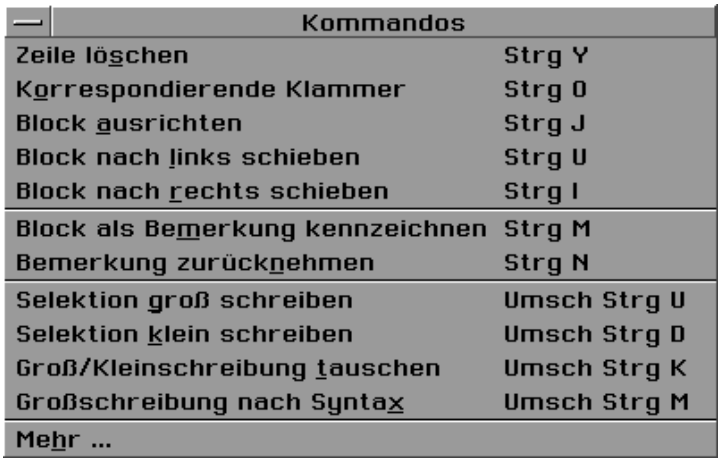
Insbesondere ist es möglich, den Quellcode oder einen Teil davon zu drucken. Dabei können Sie einige Einstellungen wählen, um den ausgedruckten Quellcode zu formatieren.
Der Punkt "Bereinigte Kopie speichern" kopiert Ihren Quelltext und alle wichtigen Daten in eine neue, leere Datei. Verlorene Handles und ungenutzter Speicher werden nicht mit kopiert.



Das Bearbeiten-Menü

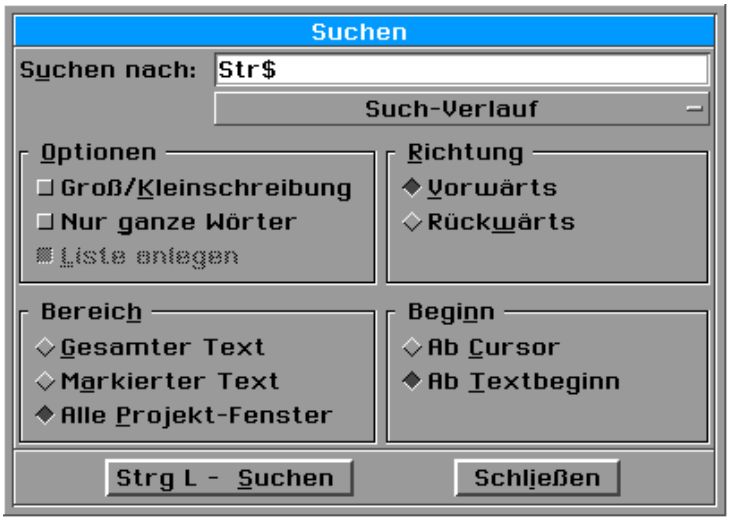


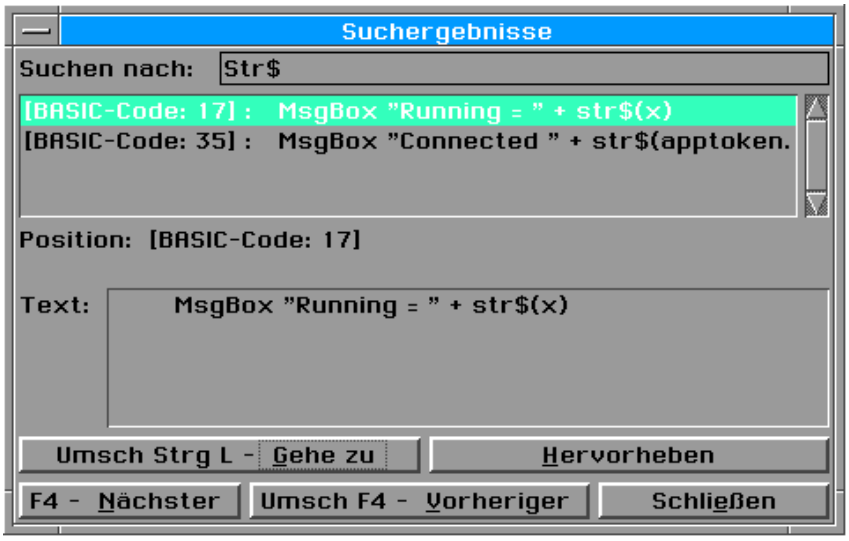
Im Menü Bearbeiten finden Sie die Funktionen um Ihren Quelltext zu editieren. Es ist wichtig zu wissen, dass R-BASIC über eine mehr als einhundert stufige Undo-Kette verfügt. Sie können also auch größere Änderungen im Quelltext gezielt zurücknehmen.
Neben den Standard-Operationen finden Sie hier auch einige spezielle R-BASIC Funktionen. Insbesondere sei hier auf die Menüpunkte "Suche in Dateien", "Gehe zu Zeile" und "Kommandos" hingewiesen.



Die Tastatur-Kommandos erleichtern die Arbeit mit dem Editor. Eine komplette Liste der Tastatur-Kommandos finden Sie im Hilfe-Menü oder über den Menüeintrag "Mehr ...".

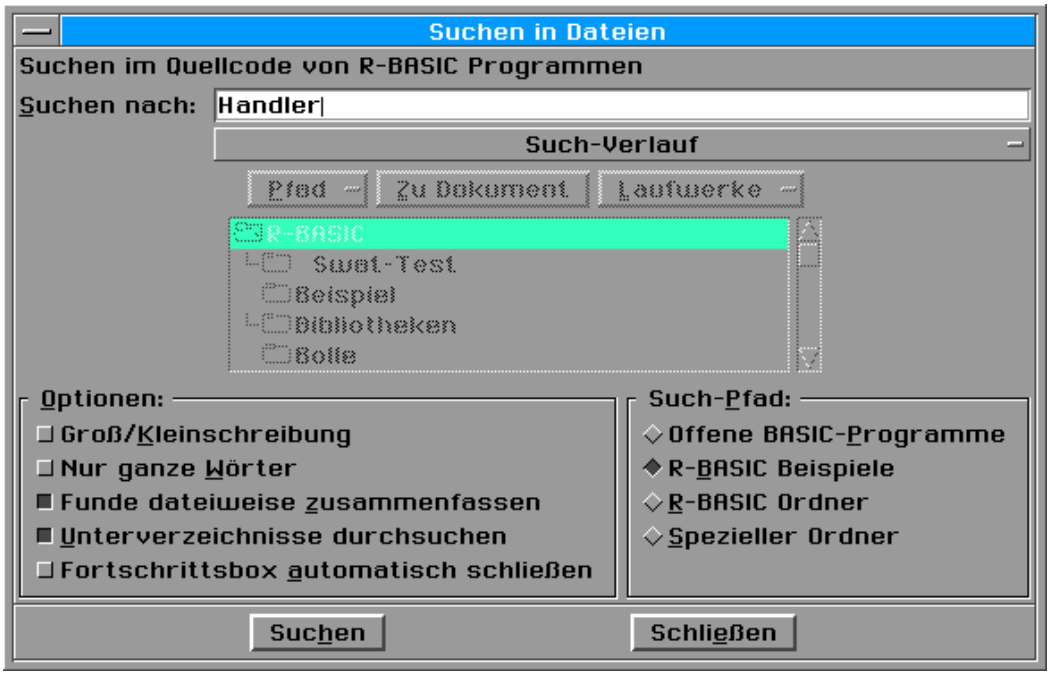
R-BASIC hat getrennte Dialogboxen für "Suchen" und "Ersetzen". Hier ist der Suchen-Dialog gezeigt.
Besonders hingewiesen werden soll hier auf den Punkt "Liste anlegen" unter "Optionen". R-BASIC erzeugt dann eine Liste aller Fundstellen, so dass Sie die Fundstellen immer wieder und in beliebiger Reihenfolge anspringen können.



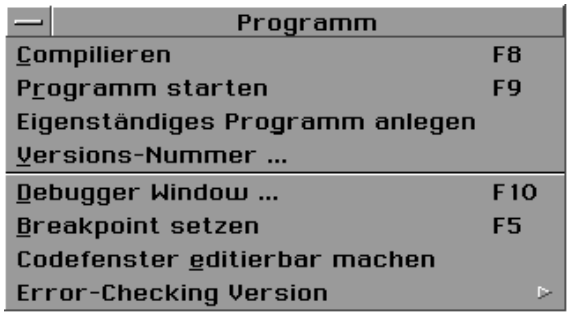


Besonders hinzuweisen ist auf die Tatsache, dass die Tastenkürzel F4 und Umsch-F4 auch dann funktionieren, wenn die Dialogbox "Suchergebnisse" nicht mehr offen ist.

Mit "Suche in Dateien" haben Sie die Möglichkeit, R-BASIC Code Dateien gezielt nach Schlagworten, z.B. nach einem Befehl, zu durchsuchen.



Das Programm-Menü



Dieses Menü enthält alle Funktionen, mit denen Sie aus ihrem Quelltext ein lauffähiges Programm erzeugen können.

Wenn Sie Ihr Programm ausprobieren wollen muss es zuerst **compiliert** werden. Dabei wird der Quelltext auf syntaktische Fehler geprüft und in eine Folge von Tokens übersetzt. Das heißt, intern wird jeder Programmbefehl durch eine Nummer ersetzt. Diese Tokens werden beim Ausführen des Programms gelesen und die entsprechende Funktion wird aufgerufen.

Nachdem das Programm compiliert wurde können Sie es **starten**. Dabei wird das Programm unter Kontrolle der IDE ausgeführt und bei Fehlern zur entsprechenden Stelle im Quelltext gesprungen. Der Menüpunkt "Programm starten" compiliert das Programm automatisch, wenn es noch nicht compiliert ist.

Mit dem Menüpunkt "**Eigenständiges Programm anlegen**" erzeugen Sie ein Programmpaket, das unabhängig von der IDE lauffähig ist. Diese "R-Apps" können auch auf dem System von Usern ausgeführt werden, die die R-BASIC IDE nicht installiert haben. Details zu diesem Menüpunkt sind im Kapitel 3 des Benutzerhandbuchs beschrieben.

Unter "**Versions-Nummer**" können Sie die Versionsnummer des Programms oder die Library ändern. Details zu diesem Thema finden Sie im Programmierhandbuch in den Kapiteln 2.11.1 (Versionsnummern) und 2.12.1 (Konzeptionelles zur Verwendung von Libraries).

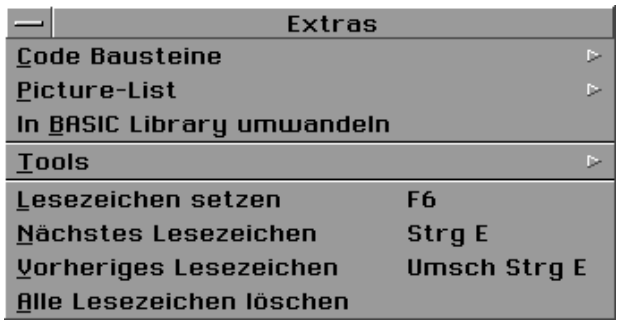
Mit "**Debugger Window**" öffnen Sie das Debugger-Fenster. Der Debugger ist ausführlich im Kapitel 5.3 dieses Handbuchs beschrieben.

Während ein Programm unter der Kontrolle der IDE läuft können Sie den Sourcecode nicht ändern. Insbesondere wenn Sie Zeilen löschen oder hinzufügen könnte der Debugger fehlerhafte Anzeigen liefern. Mit "**Codefenster editierbar machen**" können Sie diesen Schutz abschalten. Außerdem können Sie dann bei laufendem Programm mit **F5** einen **Breakpoint** in der aktuellen Zeile setzen.

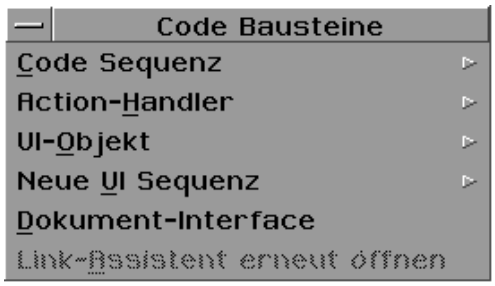
Der Menüpunkt "**Error-Checking Version**" erlaubt Ihnen einzustellen, ob Sie eine Programmversion zur Fehlersuche (EC-Version) oder die "normale" Version (NC-Version) compilieren wollen.

Dieses Thema wird im Abschnitt 5.2 dieses Handbuchs ausführlich besprochen.

Das Extras Menü



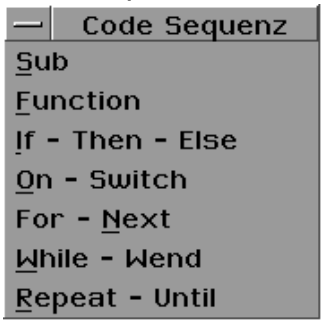
Im Extras-Menü finden Sie alles zur Arbeit an Ihrem Programm, was nicht ins Bearbeiten- oder ins Programm-Menü passt.



Code Bausteine

Das ist der leistungsfähigste Eintrag in diesem Menü. Er erspart Ihnen das Tippen ganzer Programmsequenzen. Sie sollten diese Menüpunkte gründlich ausprobieren!

• Code Sequenz

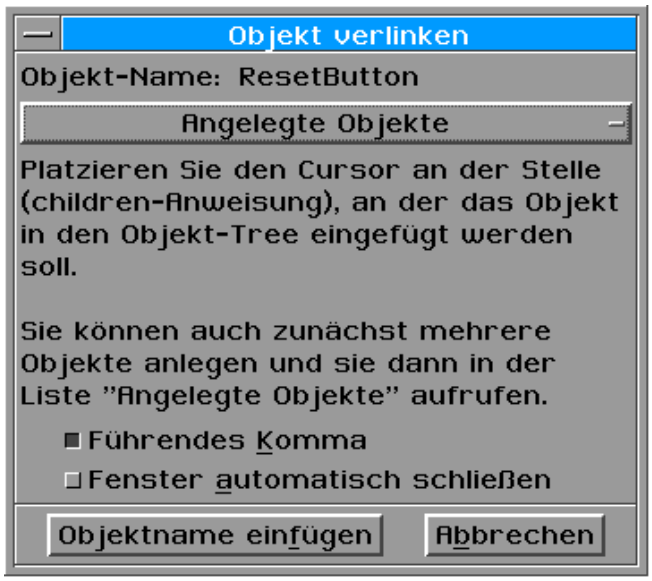


Hier sind besonders die Einträge Sub und Function zu empfehlen. Sie erzeugen eine "leere" Sub oder Function, wobei Sie Namen und Parameter eingeben können.
Analog erzeugt "**Action-Handler**" einen leeren Actionhandler. Hier ist der automatisch angelegte Kommentarblock wichtig, denn er enthält eine Beschreibung der Parameter des Handlers.

• UI-Objekt

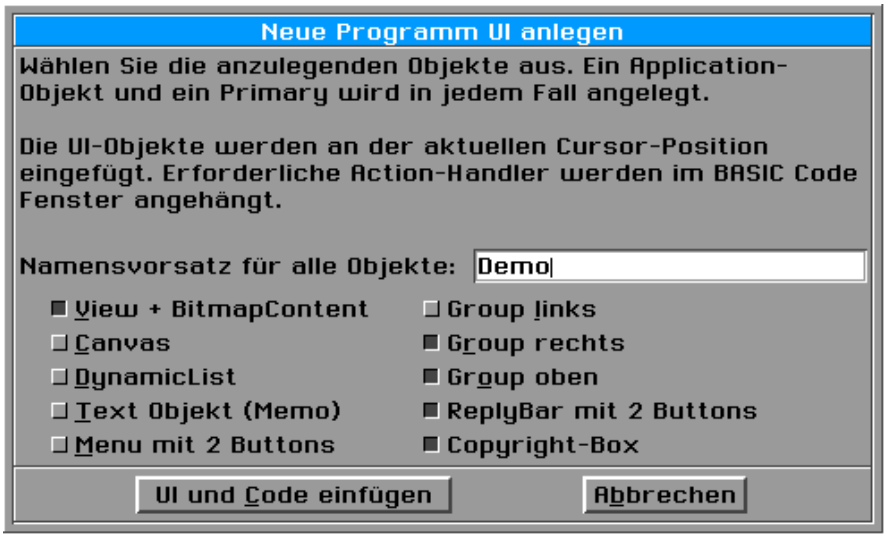
Hier finden Sie alle in R-BASIC verfügbaren Objekt-klassen. Der von diesem Menüpunkt in ihren Quelltext eingefügte UI-Code enthält als Kommentar die wichtigsten (nicht alle!) Instancevariablen des entsprechenden Objekts.

Nachdem das Objekt eingefügt wurde erscheint eine Dialogbox, die Sie an das Einbinden des neuen Objekts erinnert und Sie dabei unterstützt.



• Neue UI-Sequenz

Mit diesem Punkt fügen Sie typische Objektkonstellationen wie eine View/Content Kombination, eine RadioButtonGroup mit mehreren RadioButton-Objekten oder eine Copyright-Box in ihren UI-Code ein. Besonders hingewiesen werden soll hier auf den Unterpunkt "**Komplexe Programm UI**", mit dem Sie die Grundstruktur für sehr viele Programmtypen mit wenigen Mausklicks erzeugen können.



• Document Interface

Dieser Eintrag ist etwas für fortgeschrittene Programmierer. Er fügt Ihrem Programm mehrere Hundert Zeilen Programm- und UI-Code zu. Damit ist es mit relativ geringem zusätzlichem Aufwand möglich, eine komplettes Dokumentenverwaltung (Öffnen, Schließen, Speichern unter ...) in Ihr Programm einzubauen. Die Details zu diesem Thema sind im Handbuch "Spezielle Themen, Vol. 3, Kapitel 15, ausführlich beschrieben.

• Link-Assistent erneut öffnen

... öffnet den "Objekt verlinken" Dialog aus dem Menüpunkt "UI-Objekt". Diese Dialogbox ist auch im BASIC Code hilfreich um die exakten Namen von Objekten, die Sie gerade angelegt haben, im Code einzutragen.

Picture-List

Mit der Picture-List steht Ihnen die Möglichkeit zur Verfügung, Bilder in der Code-Datei selbst zu speichern und zur Laufzeit komfortabel zu zeichnen. Die Benutzung der Picture-List und die dazugehörigen Befehle sind im Programmierhandbuch, Vol. 3, Kapitel 2.8.6 beschrieben.

In BASIC Library umwandeln

Libraries sind Sammlungen von Routinen, Konstanten, Strukturen usw., die von mehreren Programmen gleichzeitig genutzt werden können. Im Kapitel 2.12 des Programmierhandbuchs (Vol. 4) ist beschrieben wie man eine Library schreibt. Da sich die interne Struktur einer BASIC Library etwas von der eines BASIC

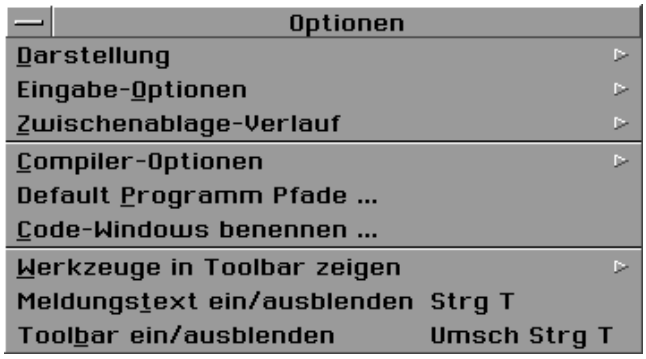
Programms unterscheidet, müssen Sie ein neues, leeres Programm anlegen und dann diesen Menüpunkt anwählen, wenn Sie eine Library erstellen wollen.

Tools
Tools sind Zusatzprogramme, die unabhängig von R-BASIC laufen, aber bei der Arbeit mit R-BASIC hilfreich sein können. R-BASIC liefert einige Tools mit, weitere können von der R-BASIC Webseite heruntergeladen werden.

Um ein Programm im "Tools" Menü verfügbar zu machen genügt es, dieses Programm oder einen Link auf dieses Programm in den Ordner "USERDATA\R-BASIC\Tools" zu kopieren. Bei jedem Start von R-BASIC wird dieser Ordner durchsucht und alle gefundenen Programme werden in das Menü eingetragen.

Lesezeichen ...
Lesezeichen sind Markierungen, die ein schnelles Auffinden von Textstellen erlauben. Sie werden im Dokument gespeichert und beziehen sich jeweils nur auf das aktuelle Code-Window.

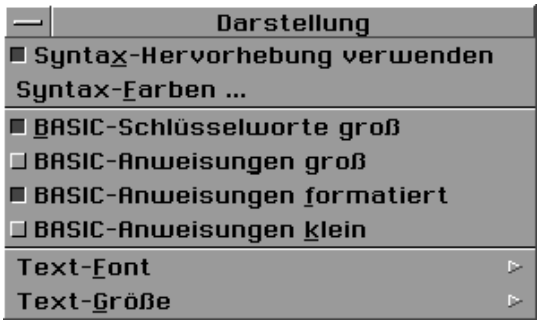
Das Menü Optionen



Der Menüpunkt "Optionen" ist in drei Gruppen geteilt. Alle Einstellungen, die Sie hier vornehmen, werden automatisch in der GEOS.INI gespeichert. Ausnahmen davon sind unten explizit erwähnt.

Die erste Gruppe erlaubt Einstellungen, die direkt bei der Eingabe ihres Quelltextes bemerkbar machen.

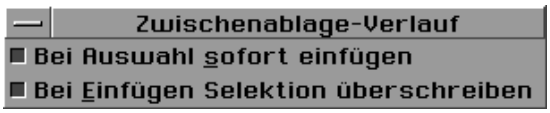
- Darstellung
  Unter diesem Punkt können Sie einstellen, wie der Quelltext dargestellt wird. Dazu gehört die Schriftart, die Schriftgröße und vor Allem, ob und in welcher Weise die Syntax-Hervorhebung verwendet wird.



• Syntax-Farben
Hier stellen Sie die für die Syntax-Hervorhebung verwendeten Farben ein. Sie können verschiedene Farbschema anlegen, diese speichern und laden.

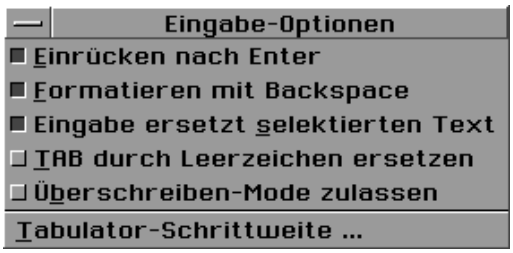
• Text-Font, Text-Größe
Hier kann man die im Editor verwendete Schriftart und Zeichengröße einstellen.

• Zwischenablage-Verlauf



Die Optionen zum Clipboard-Verlauf beziehen sich auf das Clipboard-History Werkzeug unter der Menüleiste. Die gleichen Werte kann man auch dort einstellen.

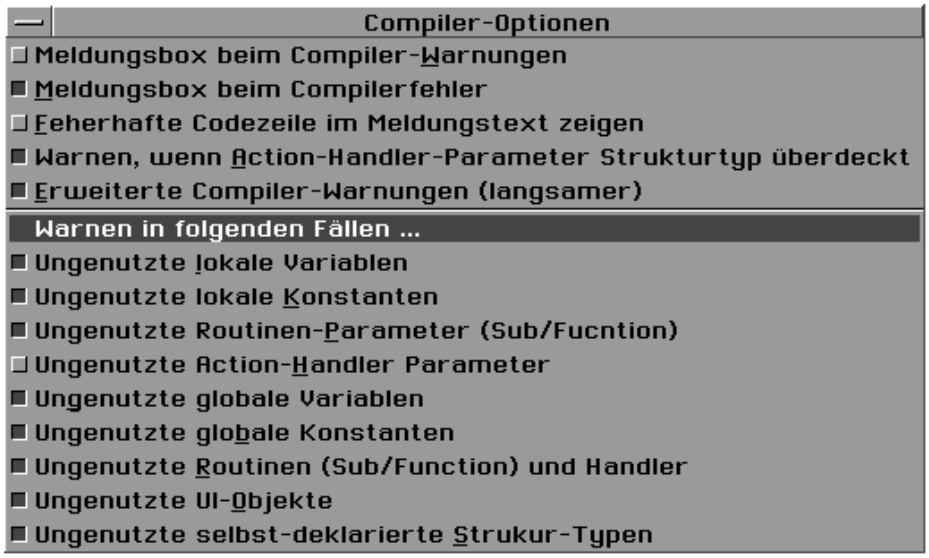
• Eingabe-Optionen
Diese Einstellungen wirken sich direkt bei der Eingabe des Quelltextes aus.



• Tabulatorschrittweite
Die Schrittweite beim Betätigen der Tabulator-Taste kann für einzelne Codefenster getrennt eingestellt werden. Sie können auch einen neuen Standardwert für neue Dokumente festlegen.

Die mittlere Gruppe beeinflusst das Verhalten des Compilers und ermöglicht Ihnen, den Code-Windows an Ihr Projekt angepasste Namen zu geben.

• Compiler-Optionen



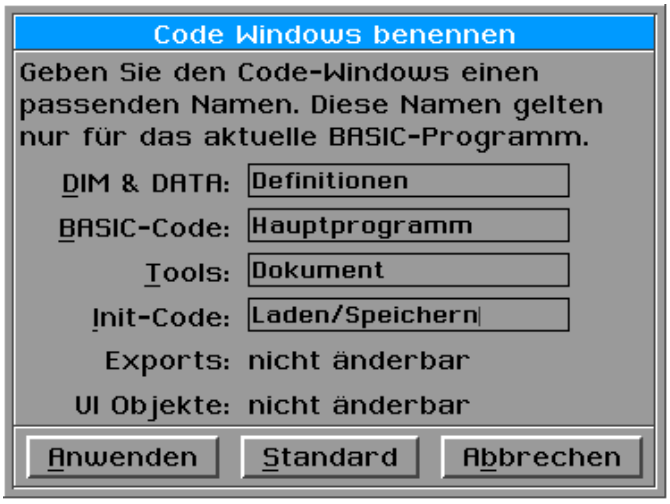
Hier bedarf die Option "Warnen, wenn Action-Handler-Parameter Strukturtyp überdeckt" einer Erklärung. Da bei Action-Handlern die Parameter im Code nicht aufgeführt werden, kann die Situation auftreten, dass Sie einen Strukturtyp definiert haben, dessen Bezeichnung mit dem Namen eines der Parameter des Handlers übereinstimmt, ohne dass Sie dies sofort bemerken. Sie können dann im Handler keine lokalen Variablen von diesem Typ definieren. Aktivieren Sie diese Option, damit der Compiler diese seltene Situation erkennt und eine Warnung ausgibt.

• Default Programm Pfade

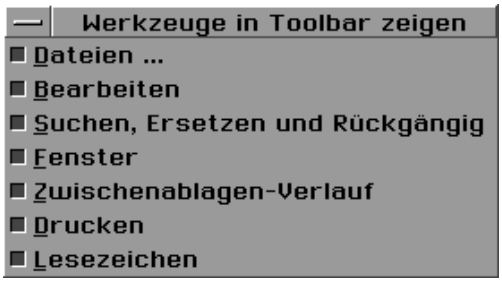
In dieser Dialogbox legen Sie fest, wo bestimmte Dateien standardmäßig abgelegt werden. Die letzten drei Einträge sind die Vorgabewerte für den Menüpunkt "Eigenständiges Programm anlegen" aus dem "Programm"-Menü. Sie können die Pfade dort noch an die Erfordernisse des aktuellen Projekts anpassen.

• Code Windows benennen

Hier können Sie die Bezeichnungen von 4 der sechs Code-Windows ändern. Die neuen Namen werden in der Code-Datei gespeichert. So können Sie für jedes Programmier-projekt eigene Namen verwenden. Code-Windows, die eine Sonderfunktion haben, können nicht umbenannt werden.



Die letzte Gruppe beeinflusst die in der IDE angezeigten UI-Objekte.



• Werkzeuge in der Toolbar zeigen
Hier können Sie die einzelnen Elemente unabhängig voneinander ein- und ausblenden.

• Meldungstext ein/ausblenden
Mit diesem Menüpunkt oder der Tastenkombination Strg-T können Sie den Bereich unter dem Quelltext minimieren, so dass Sie mehr Platz zur Eingabe des Quelltextes haben. Diese Einstellung wird nicht in der GEOS.INI gespeichert. Beim Compilieren wird der Meldungstext automatisch wieder eingeblendet.

• Toolbar ein/ausblenden
Wer noch mehr Platz braucht kann mit Shift-Strg-T auch die Toolbar temporär ausblenden. Mit Ausnahme des Zwischenablagenverlaufs sind alle Funktionen weiterhin über die Menüs oder ihre Tastenkombinationen erreichbar. Auch diese Einstellung wird nicht in der GEOS.INI gespeichert.

Das Fenster Menü

Dies ist ein Standardmenü, das in allen Anwendungen, die mit Dokumenten arbeiten, vorkommt. In R-BASIC gibt es hier keine Besonderheiten.

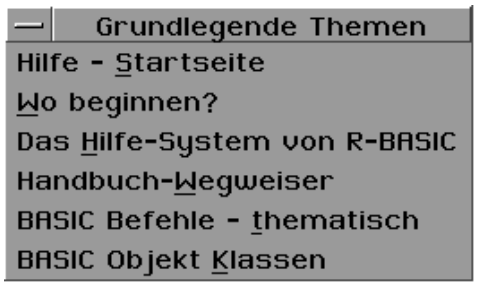
Das Hilfe Menü

Das Hilfe-Menü ist der Startpunkt, wenn Sie nach einem bestimmten Befehl oder der Beschreibung zu einem Befehl suchen.

Die **Wizzards** sind der Kern der BASIC Hilfe. Der **Code-Wizzard** enthält eine Kurzbeschreibung aller BASIC Befehle, Konstanten Strukturen usw. Der **Objekt Klassen Wizzard** enthält eine Kurzbeschreibung alle Objektklassen, ihrer Instancevariablen und der dazugehörigen Konstanten. Die Wizzards der Libraries (z.B. der KeyCodes Library) sind ebenfalls erreichbar, wenn Sie den BASIC-Code Wizzard öffnen. Mit der F2-Taste (**Quick-Help**) werden alle Wizzards nach dem Identifier an der Cursorposition durchsucht und der entsprechenden Eintrag wird angezeigt.

Der Eintrag "**Grundlegende Themen**" verweist auf wichtige Kapitel aus der Hilfedatei. Die R-BASIC Hilfedatei enthält drei wichtige Elemente:

1. Sie ist Wegweiser durch die Handbücher.
2. Sie enthält eine thematische Übersicht und Kurzbeschreibung aller BASIC-Befehle.
3. Sie enthält eine Kurzbeschreibung aller Objektklassen und der dazugehörigen Instancevariablen.



Unter "**R-BASIC Handbücher**" finden Sie eine Liste aller Handbücher und Dokumente aus dem Ordner "R-BASIC\Dokumentation" sowie deren Inhaltsverzeichnisse. Wenn Sie das passende Thema gefunden haben können Sie das entsprechende Handbuch durch einen einfachen Mausklick direkt öffnen.

Mit "**Handbücher durchsuchen**" starten Sie eine Volltext-Suche in den Handbüchern im Ordner "R-BASIC\Dokumentation".

Der Eintrag "**Tastaturbefehle**" verweist auf eine Seite aus der Hilfedatei. Hier finden Sie eine komplette Liste alle Tastaturkommandos, die im Editor zur Verfügung stehen.

(Leerseite)

3 Eigenständige Programme anlegen

Wenn Sie ein Programm fertiggestellt haben möchten Sie es sicher mit anderen Nutzern teilen. Darunter sind sicher auch Nutzer, die R-BASIC gar nicht installiert haben und dies auch nicht wollen. Es ist deswegen möglich ein compiliertes Programm so zu "exportieren" dass es unabhängig von der R-BASIC Entwicklungsumgebung lauffähig ist. Nach außen verhält sich ein solches "eigenständiges Programm" (auch R-App genannt) wie ein "normales" mit dem PC/GEOS-SDK geschriebenes Programm. Es hat z.B. ein eigenes Icon, das der Nutzer im WORLD Verzeichnis sieht und über das er das Programm starten kann.

Außerdem erstellt R-BASIC bei Bedarf ein Uni-Installer Paket, das automatisch alle Dateien enthält, die für die Verwendung des Programms notwendig sind.

3.1 Interne Organisation

Ein R-BASIC Programm, das unabhängig von der Entwicklungsumgebung laufen soll, besteht aus 4 Teilen:

1. Die Binär-Datei (BIN-Datei)
   Diese Datei enthält den eigentlichen compilierten BASIC Code und befindet sich im Ordner USERDATA\R-BASIC\BIN bzw. einer seiner Unterordner.

2. Der Launcher
   Der Launcher (deutsch: Starter) ist ein kleines SDK-Programm, dass sich im Ordner WORD oder einer seiner Unterordner befindet. Er ist die Schnittstelle zwischen Nutzer und BASIC Programm.

3. Libraries
   Zum Ausführen eines BASIC Programms benötigt der Launcher diverse Libraries aus dem SYSTEM Ordner (z.B. die R-BASIC Runtime Library), die BASIC Library "BasicSystemTypes" sowie gegebenenfalls die vom Programm includeten R-BASIC Libraries, die sich im Ordner "USERDATA\RBASIC\-Library\BIN" befinden.

4. Weitere Dateien
   Möglicherweise werden weitere Dateien benötigt. Das können z.B. Hilfedateien für das Programm sein oder Dateien, irgendwelche Daten enthalten.

Legen Sie ein eigenständiges Programm an, so passiert folgendes:
1. R-BASIC erzeugt die BIN Datei, indem es eine Kopie des aktuellen BASIC Programms in USERDATA\R-BASIC\BIN\... ablegt. Die Kopie wird dann modifiziert, z.B. wird die Release-Nummer gesetzt.
2. R-BASIC legt eine Kopie des Default-Launchers im WORLD Ordner ab. Diese Kopie wird dann modifiziert ("gepatcht"), z.B. werden der Name, die Tokens (einschließlich der dazugehörigen Bildchen) und der Pfad zur BIN Datei in die Datei geschrieben. Damit existiert eine eindeutige Zuordnung zwischen Launcher und BIN Datei.

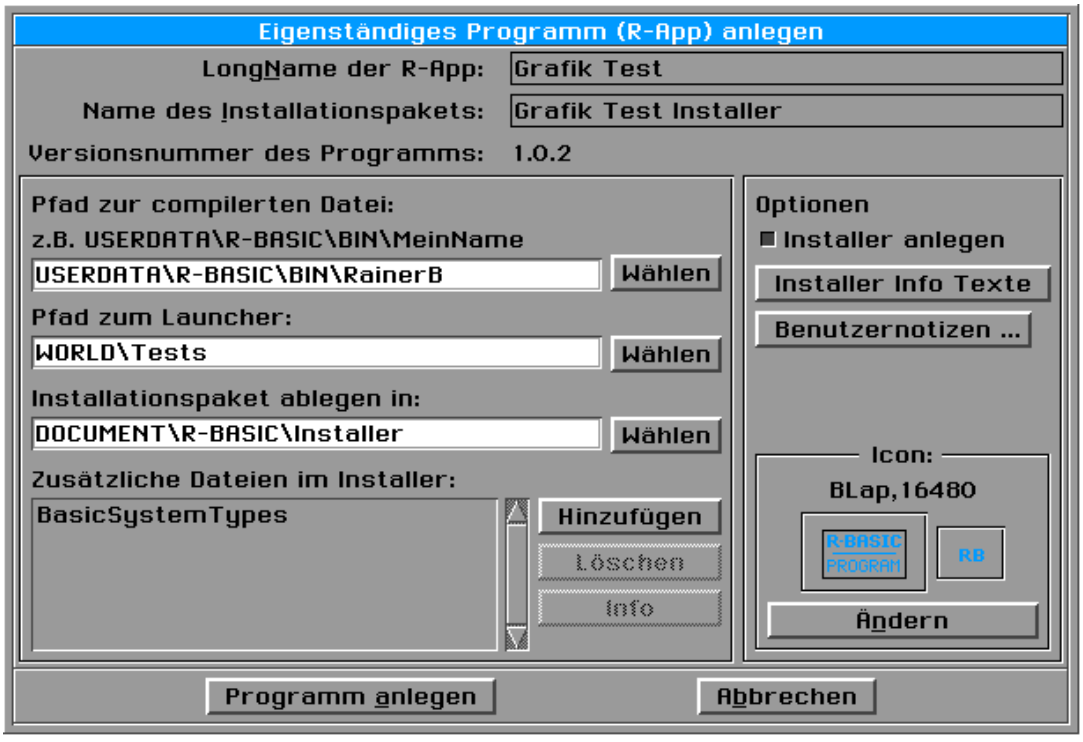
3. Wenn gewünscht wird ein Uni-Installer Paket angelegt, das alle zur Installation des Programms auf anderen Systemen nötigen Dateien enthält.

Der oben beschriebene Prozess bewirkt auch, dass ein nachfolgendes Ändern und erneutes Compilieren ihres Programms das eben erstelle eigenständige Programm nicht beeinflusst. Weder die BIN Datei noch der Launcher werden durch den Compileprozess geändert. Sollten Sie eigene BASIC Libraries includen und compilieren Sie eine davon neu, so wird das Programm automatisch die neue Version der Library verwenden. In diesem Fall sollten Sie auch das Installationspaket mit Hilfe des Uni-Install Creators updaten.

Es wird empfohlen beim Anlegen des eigenständigen Programms auch ein Uni-Installer-Paket durch R-BASIC anlegen zu lassen. Dieses enthält automatisch alle benötigten Dateien (Punkte 1 bis 3) und das Einbinden weiterer Dateien (Punkt 4) wird komfortabel unterstützt. Die benötigten System-Libraries werden aus Ihrem SYSTEM Ordner in das Installer-Paket kopiert. Dadurch enthält das Paket immer die aktuellen Library-Versionen und Kompatibilitätsprobleme sind ausgeschlossen. Der Uni-Installer sorgt beim Installationsprozess automatisch dafür, dass neuere Libraries nicht mit älteren Versionen überschrieben werden.

Startet der Nutzer nach Installation des Programms den Launcher, so liest dieser die BIN Datei und führt das eigentliche BASIC Programm aus. Da der Launcher ein PC/GEOS-SDK Programm ist greifen hier die systemweiten Kompatibilitätschecks (z.B. auf fehlende oder inkompatible Libraries) und auch die Installation der Programm-Token (Icon Bildchen) läuft genau wie bei einem mit dem PC/GEOS-SDK geschriebenen Programm automatisch ab.

3.2 Der Dialog zum Programm anlegen



LongName des Programms

Der "LongName" ist der unter PC/GEOS sichtbare "lange" Dateiname. In den meisten Fällen ist er identisch mit dem Namen ihrer R-BASIC Programm Datei. Sie können bei Bedarf aber einen abweichenden Namen spezifizieren - entweder hier oder über die Instancevariable **LongName$** des Application Objekts. Der LongName wird sowohl für den Launcher als auch für die BIN Datei verwendet.

Name des Installationspakets

Häufig soll das Installationspakt nicht exakt genauso heißen wie das Programm. Heißt ihr Programm z.B. "SuperGame" könnte das Installationspaket "SuperGame Version 1.0" oder "SuperGame Version 1.1" heißen. Beim Upgrade von Version 1.0 auf 1.1 hat der Nutzer dann nicht zwei Supergames in seinem World Ordner.

Versionsnummer des Programms

Die Versionsnummer des Programms wird über den Menüpunkt "Programm" "Versionsnummer" eingestellt. Der letzte der drei Werte wird bei jedem Compilervorgang automatisch hochgezählt. Der Uni-Installer stellt über die Versionsnummer des Programms sicher, dass niemals eine ältere Programmversion eine neuere überschreibt.

Pfad zur compilierten Datei

Hier wird die BIN Datei abgelegt. Es wird empfohlen, dass Sie sich unter R-BASIC\BIN einen Unterordner erstellen, der ihrem Namen entspricht und diesen verwenden. Dort werden dann alle ihre BIN Dateien abgelegt und Sie können bei Bedarf hier auch weitere für ihre Programme benötigte Dateien unterbringen (z.B. Spiele Levels), ohne dass Sie befürchten müssen, dass es zu Namenüberschneidungen mit den Programmen anderer Programmierer kommt.

Über den Menüpunkt "Optionen" - "BASIC Optionen" - "Default Programm-pfade" können Sie einen Startwert setzen, der für alle neuen Projekte automatisch übernommen wird.

Pfad zum Launcher

Installationspaket anlegen in

Diese Einträge sollten selbsterklärend sein. Auch hier können Sie über den Menüpunkt "Optionen" - "BASIC Optionen" - "Default Programmpfade" einen Startwert setzen, der für alle neuen Projekte automatisch übernommen wird.

Zusätzliche Dateien im Installer

Hier erscheinen automatisch alle über das Schlüsselwort INCLUDE eingebundenen R-BASIC und SDK-Libraries. Dazu kommt die Library "BasicSystemTypes". Sollte eine Library weitere Libraries includen, so erscheinen diese hier auch automatisch. Zusätzlich können Sie weitere Dateien zur Liste hinzufügen, z.B. Hilfedateien, Grafikdateien oder andere Daten-Dateien. Alle in dieser Liste enthaltenen Dateien werden automatisch in das Installationspaket übernommen.

Hinweis: Um dem Löschen von wichtigen Dateien (z.B. Fonts) beim Deinstallieren des Paktes vorzubeugen, sind alle in dieser Liste vereinbarten Dateien als "Beim Deinstallieren nicht löschen" markiert. Wenn Sie das nicht wollen, können Sie das Paket anschließend mit der Uni-Install-Creator Applikation bearbeiten.

Optionen

- ***Installer anlegen***
  R-BASIC wird nach erfolgreichem Erstellen des eigenständigen Programms alle benötigten Dateien in ein Uni-Installer Paket schreiben. Das Paket ist danach sofort einsetzbar, Sie können es aber mit dem Uni-Install Creator noch modifizieren.

Installer Info Texte

Erlaubt die Eingabe der Informationstexte, die der Uni-Installer dem Nutzer präsentiert.

Icon

Spezifiziert das Icon, dass der Launcher im GeoManager haben soll. Wenn Sie kein AppToken im Application-Objekt spezifiziert haben wir das oben gezeigte Standard Token verwenden. Sie können das Token hier noch einmal ändern, bevor sie das eigenständige Programm anlegen.

Klicken Sie, nachdem sie alle Einstellungen getätigt haben, auf "Programm anlegen" und R-BASIC macht die Arbeit für Sie.

Alle Daten, die in diesem Dialog eingegeben werden, werden beim Erstellen des eigenständigen Programms im originalen R-BASIC Programm abgespeichert. Damit stehen sie wieder zur Verfügung, wenn Sie eine neue Version Ihres Programms erzeugen wollen. Eine Ausnahme bilden Werte, die im UI-Code spezifiziert wurden (z.B. LongName$ und AppToken). Diese werden, wenn sie im UI Code stehen, bei jedem Compilevorgang neu gesetzt.

3.3 Verwendung von eigenen Icons

Die Begriffe Token und Icon werden sowohl bei den Anwendern als auch im umgangssprachlichen Gebrauch der Programmierer nicht sauber getrennt, auch nicht in diesem Handbuch. Strenggenommen versteht man unter "Token" eine Struktur aus 4 Buchstaben (den TokenChars) und einer 16-Bit Zahl, der Manufacturer-ID. Der Name dieser Struktur ist **GeodeToken**. Unter R-BASIC ist sie 7 Byte groß, weil die Zeichenkette mit einer Null abgeschlossen werden muss. Im PC/GEOS-SDK sind es nur 6 Byte. Zu jedem Token gehört ein ganzer Satz von Bildern, den "Icons". Jedes Icon enthält Informationen, für welchen Zweck es sich am besten eignet, z.B. als Icon im GeoManager (Kennung "Icon"), als Mini-Bildchen für das Expressmenü (Kennung "Tool"), für welche Bildschirmauflösung es geeignet ist, welche Farbtiefe es hat (monochrom, 16 oder 256 Farben) und noch einige mehr. Je nach konkreter Situation wählt GEOS aus den vorhandenen Bildern das passende aus. Ist kein passendes vorhanden wählt es ein "möglichst gut passendes".

R-BASIC unterstützt genau zwei Icons (Bilder) pro Token: Das eigentliche Icon ist 48x30 Pixel groß, vom Typ "Icon" und hat 16 oder 256 Farben. Es wird vom GeoManager bei der Anzeige des Programms verwendet. Das Andere ist 15x15 Pixel groß, vom Typ "Tool" und kann ebenfalls 16 oder 265 Farben haben. Es wird z.B. für das Expressmenü benutzt. Die Verwendung von Token ohne das 15x15 Tool Icon ist möglich, aber nicht zu empfehlen. Geringfügige Abweichungen von den vorgegebenen Größen sind zulässig, sollten aber mit Bedacht eingesetzt werden.

Die Zuweisung eines Icons zu Ihrem R-BASIC Programm ist sehr einfach. Verwenden Sie im UI Code das Statement **AppToken** für Ihr Application-Objekt:

```
Application  MyApp
  Children = MyPrimary
  AppToken = "BPdm",16600
END OBJECT
```

Alternativ können Sie beim Anlegen des Programms ein Token aus Ihrer TokenDB (Token Database Datei) auswählen.

Beim Compilieren des Programms merkt sich der Compiler das Token und beim Anlegen des eigenständigen Programms kopiert R-BASIC das Token einschließlich der dazugehörigen Bilder in den Launcher. Einzige Bedingung ist, dass sich das Token in **Ihrer** TokenDB Datei befinden muss. Verwendet nun ein Nutzer ihr R-BASIC Programm und das Token befindet sich noch nicht in seiner TokenDB, dann installiert der Launcher das Token automatisch in der TokenDB Datei des Nutzers, genauso wie es ein "normales" PC/GEOS-SDK-Programm macht.

Mit den Statements **DocToken** und **ExtraToken** können Sie zwei weitere Tokens definieren, die aus der TokenDB in den Launcher kopiert werden und die der Launcher bei Bedarf gemeinsam mit dem AppToken in die TokenDB des Nutzers installiert. Dabei werden die GEOS Systemfunktionen benutzt. Das bedeutet:

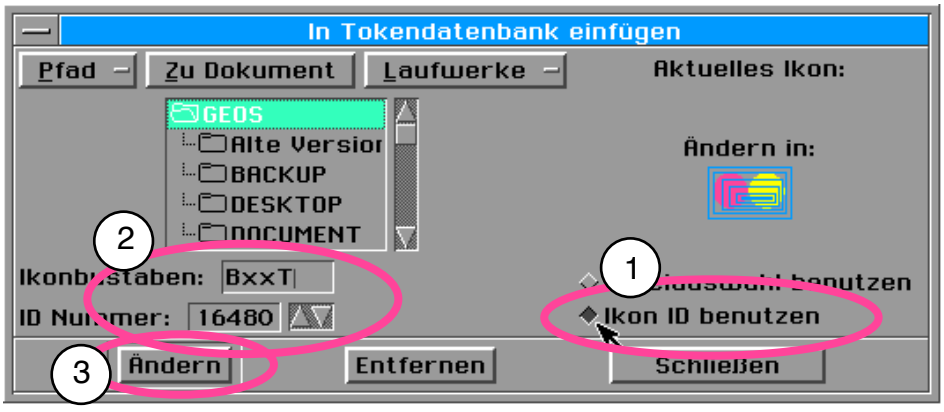
findet der GeoManager bei der Anzeige des Programms (Launchers) das AppToken nicht seiner TokenDB, so wird der Launcher aufgefordert, seine Token zu installieren. er installiert dann AppToken und - falls vorhanden - DocToken und ExtraToken. Ist das AppToken bereits in der TokenDB wird nichts installiert. DocToken und ExtraToken werden nicht extra geprüft.

Ein eigenes Icon erstellen

Die im Folgenden benannten Programme sind im Netz, unter anderem auf der R-BASIC Seite, zu finden.

Der einfachste Weg ein eigenes Icon zu erstellen ist die Verwendung des Programms "**Icon Editor**". Verwenden Sie immer die aktuellste Version des Icon Editors! Ältere Versionen enthalten diverse Bugs und sind z.T. instabil.
Wenn Sie ein selbst gezeichnetes Icon verwenden wollen benötigen Sie eine eigene Manufacturer-ID. Es ist sehr leicht eine eigene Manufacturer-ID zu bekommen. Details dazu sind im Abschnitt 3.4 "Über die Manufacturer-ID" beschrieben. Sie sollten für jedes R-BASIC Token genau zwei Bilder erstellen (Menüpunkt "Format", "Format hinzufügen"): ein 48x30 Pixel großes mit den Eigenschaften: "VGA", "Icon" und "Standard" sowie ein 15x15 Pixel großes mit den Eigenschaften: "VGA", "Werkzeug (Tool)" und "Klein (Tiny)".

Der Icon Editor verfügt über eine Importfunktion aus der TokenDB. Sie können also existierende Tokens importieren, ggf. ändern, ein 15x15 Tiny-Icon hinzufügen und das modifizierte Token wieder in die TokenDB exportieren. Dabei sollen Sie auf jeden Fall Ihre eigene Manufacturer-ID verwenden (und ggf. die TokenChars ändern). Ändern Sie die Tokenkennung nicht kann es passieren, dass Ihr geändertes Icon bei anderen Nutzern nicht anzeigt wird, weil Sie das Original bereits in ihrer eigene TokenDB haben.



Um ein neues Token in die TokenDB zu exportieren wählen Sie unter "Datei" den Menüpunkt "In Tokendatenbank einfügen". Dort **müssen** Sie unbedingt die Auswahl "Icon ID benutzen" treffen (1). Je nach Version des Iconeditors kann die Beschriftung der einzelnen Punkte variieren. Außerdem müssen Sie eine ID Nummer und die TokenChars (exakt 4 Buchstaben oder Ziffern) eingeben (2). kann können Sie auf "Ändern" klicken (3).

Als Quelle für Icons eignen sich z.B. Iconsammlungen. Eine große Anzahl davon ist auf der Geos-Infobase, unter "Download Programme" - "Icons" zu finden. Diese liegen als TokenPaks (erfordert zur Installation den "TokenLoader") oder als Sammlung von "Token-Dummys" vor. Dies sind kleine Programme, deren einziger Zweck darin besteht ihre Tokens in die TokenDB zu schreiben.

Eine andere Möglichkeit besteht in der Verwendung des DOS Programms **ICONMAKE** von Marcus Groeber. Dieses Programm erzeugt Token-Dummys aus PCX- oder ICO-Dateien, allerdings maximal mit 16 Farben. Weitere Details sowie die unterstützten Formate sind in der dazugehörigen Dokumentation beschrieben. Einmal im GeoManager angezeigt befinden sich die Tokens dann permanent in der TokenDB und können bei Bedarf auch mit dem Icon Editor geändert bzw. um ein 15x15 "Tiny" "Tool" Icon ergänzt werden.

Zum Erzeugen von 256 Farb-Icons eignet sich das "**Token PCX Tool**". Dieses einfache Programm liest 256-Farb PCX Dateien und erstellt daraus ein Token mit einem 48x30 und einem 15x15 Bild in der TokenDB.

3.4 Über die Manufacturer-ID

Was ist eine Manufacturer-ID?

Manufacturer-ID bedeutet übersetzt so viel wie "Hersteller-Kennung" und ist eine Zahl zwischen 0 und 65535. Sie ist z.B. Teil des "Tokens" von Programmen und kennzeichnet die Person oder die Firma, die das Programm geschrieben hat. Damit soll verhindert werden, dass zwei Programme das gleiche Token haben oder es Konflikte bei anderen Systemfunktionen, wie z.B. der Arbeit mit dem Clipboard, gibt. In einer idealen Welt hat jeder Programmierer, egal ob R-BASIC oder PC/GEOS-SDK, eine eigene Manufacturer-ID und alles ist gut.

Wann brauche ich eine eigene Manufacturer-ID?

Wahrscheinlich werden Sie erstmalig eine eigene ID benötigen, wenn Sie ein Programm veröffentlichen und ihm ein selbstgezeichnetes Icon geben wollen. Es gibt aber weitere R-BASIC Funktionen (z.B. spezielle Clipboardfunktionen), die die Verwendung einer eigenen ID erfordern. Das wird bei der Beschreibung der entsprechenden Funktion deutlich dargestellt. Zum "Probieren" können Sie die ID 16600 verwenden. Diese ID ist für R-BASIC Beispiele reserviert.

Wie bekomme ich eine Manufacturer-ID ?

Es ist sehr einfach eine eigene Manufacturer-ID zu bekommen. Die ID's werden unter www.geos-sdk.com online verwaltet und vergeben. Melden Sie sich dort an und fordern Sie eine ID an. Sie können die gleiche ID für alle Ihre GEOS Projekte, egal ob R-BASIC Programm, Iconsammlung oder PC/GEOS-SDK Programm, verwenden. Einen entsprechenden Link sowie weitere bzw. aktuelle Informationen erhalten Sie auch auf der R-BASIC Homepage unter www.rbettsteller.de.

Wenn ich nun aber trotzdem keine eigene ID möchte?

Solange Sie nur für sich selbst programmieren und nichts veröffentlichen, benötigen Sie keine eigene ID. Falls Sie für Ihr Programm ein Token aus der TokenDB verwenden, das bereits eine ID hat, benötigen Sie ebenfalls keine eigene ID.
Über die ID-Bereiche sollten Sie folgendes wissen:

- Die Null ist tabu. Das ist die ID der Standard Clipboardformate und der Standardprogramme wie GeoWrite, GeoDraw usw.. Die Verwendung der Null als eigene ID ist hochgefährlich!
- ID's von 1 bis ca. 1000 werden von diversen älteren Programmen und von Iconsammlungen verwendet.
- Werte unter 16384 sind von BreadBox reserviert.
- Werte im Bereich ab 4000hex (16384 aufwärts) sind tabu. In diesem Bereich liegen die ID's der bekannten und neuen GEOS Programmierer.

Zum "Probieren" sollten Sie daher die für R-BASIC Beispiele reservierte ID **16600** verwenden. Bedenken Sie aber, dass auch andere Programmierer auf diese Idee kommen können.
Zur FormatNummer bei eigenen Clipboardformaten mit der ID 16600:

- Nehmen Sie keine Werte unter 100. Hier liegen die R-BASIC Beispiele.
- Benutzen Sie einen Zufallsgenerator.

Sobald Sie für ein veröffentlichtes Programm ein selbst erstelltes Token oder eine BASIC-Funktion verwenden, die eine eigene ID erfordert, sollte Sie sich unbedingt eine ID geben lassen.

Welche Konflikte könnten auftreten?

Die Manufacturer-ID wird unter R-BASIC genau für zwei Zwecke verwendet: Zur Definition eigener Clipboardformate und zur Definition eigener Token. Es könnte also zwei Programme geben, die das gleiche Token haben. Diese beiden Programme können niemals gleichzeitig laufen. Oder es gibt Konflikte mit dem Clipboardformat, d.h. die Kennzeichnung der Daten im Clipboard ist identisch, die Daten selbst sind aber inkompatibel. Dann kann es zum Crash beim Zugriff auf das Clipboard kommen. Mit PC/GEOS-SDK-Programmen können weitere Konflikte auftreten, da z.B. die Kommunikation zwischen Programmen oder bestimmte Systemservices über die ID abgewickelt werden.

(Leerseite)