De Re BASIC!

Version 1.68 - deutsch

Übersetzung ins Deutsche von Konrad Kellner

November 2012

Inhaltsverzeichnis

Über den Titel, De Re BASIC!

De Re ist lateinisch und bedeutet ”für die Sache”.

BASIC! Operation

Editor

Bearbeiten des Programms

Im Editor werden die BASIC! Programme geschrieben und verändert. Die Bedienung des Editors ist recht einfach.

Tippen Sie auf den Bildschirm an die Stelle, wo Sie wollen, um das Programm zu

bearbeiten. Ein Cursor erscheint. Verwenden Sie die Tastatur um an der Cursorposition zu arbeiten. Wenn das Programm, das Sie bearbeiten einen Namen durch Speichern oder Laden bekommen hat, dann wird dieser Programmname in der Titelleiste angezeigt.
Wenn Ihr Android- Gerät nicht über eine physische Tastatur verfügt, sehen Sie eine virtuelle Tastatur. Ist dies der Fall, bestehen je nach der Art und Weise wie Sie das Gerät halten, verschiedene Möglichkeiten. Wenn Sie das Gerät im Landscape-Modus halten, dann wird nur ein kleiner Bereich als Dialogfeld für den Texteingabebereich zu sehen sein. Sie können durch den Bildlauf nach oben und unten den Text in diesem Bereich bewegen, aber Sie werden vielleicht nicht in der Lage sein, sehr viel von dem Programm zu einem beliebigen Zeitpunkt zu sehen. Es ist dann möglicherweise besser nicht zu versuchen ein Programm in diesem Modus zu bearbeiten. Halten Sie dann Ihr Gerät besser im Hochformat.
Mit einem langen Berührungsdruck auf dem Bildschirm wird ein Dialogfeld angezeigt. Sie können in der Auswahlbox unter anderem Auswählen, Kopieren, Ausschneiden und Einfügen von Text auswählen. Andere Geräte haben unterschiedliche Verfahren für den Aufruf der Auschneiden- und Einfügen- Funktion.

Zeilenfortsetzung

Auf eine physikalische BASIC!- Quellcodezeile kann auf mehrere physikalische Zeilen mit dem Zeilenfortsetzungszeichen ”_” geschrieben werden
Beispielsweise könnte die Codezeile:
s$ = ”Der schnelle braune Fuchs” + Verb$ ”over” + Graf$ + ”faule Hunde”
wie folgt geschrieben werden:

```
s$ = ”Der schnelle braune Fuchs” + _
Verb$ + _
”over” + _
Graf$ + _
”faule Hunde”
```

- Formatzeile

Wenn eine Zeile das #-Zeichen am Ende der Zeile hat, werden sobald die Enter-Taste für die Zeile gedrückt wird, die Schlüsselwörter in diesem Linie großgeschrieben und das #-Zeichen entfernt. Dieses Feature funktioniert möglicherweise nicht, wenn Sie eine Soft-Tastatur verwenden. Dieses Feature funktioniert auch nicht, wenn die Einstellung ”Editor AutoIndent,” nicht aktiviert ist.

Menüs
Drücken Sie die Menütaste, um auf die folgenden Menüs zugreifen zu können.

Run
Führt das aktuelle Programm aus.
Wenn das Programm geändert wurde, seit es zuletzt gespeichert wurde, wird Ihnen hier Gelegenheit gegeben es noch zu speichern, bevor es gestartet wird.
Wenn ein Programm- Laufzeitfehler auftritt, dann wird die beanstandete Zeile ausgewählt und im Editor angezeigt. Wenn ein Programm zu schnell (weniger als 10 Sekunden nach dem Ende der vorherigen Ausführung) neu ausgeführt wir, können manchmal seltsame Laufzeit Fehler auftreten. Wenn Laufzeit Fehler auftreten, die keinen Sinn machen, versuchen Sie es erneut und warten Sie etwas länger, bevor sie es erneute Ausführen.

Load
Load wird verwendet, um eine Programm-Datei in den Editor laden. Die Programme müssen im Verzeichnis ”/sdcard/rfo-basic/source”, oder einem seiner Unterverzeichnisse stehen. Programm-Dateien müssen die Erweiterung ”. bas” haben.
BASIC! überprüft, ob das aktuelle Programm im Editor geändert worden ist, wenn Load gedrückt wird. Damit wird die Möglichkeit geboten das Programm zu speichern, wenn es geändert wurde, bevor es neu gestartet wird. Mit Save können Sie das Programm speichern. Der ”BASIC! Datei laden ”-Bildschirm zeigt eine sortierte Liste von . bas Dateien und Verzeichnissen. Verzeichnisse sind bezeichnet durch ein (d) das zu dem Verzeichnis angehängt wurde. Verzeichnis-Einträge stehen am Anfang der Liste. BASIC! Programme werden mit dem. bas Erweiterung angezeigt. Wenn sich Dateien in ”/scources/” befinden, die keine . bas-Endung haben, werden sie nicht in der Liste angezeigt.
Tippen Sie auf eine . bas Datei, um sie in den Editor zu laden.
Tippen Sie auf ein Verzeichnis, um den Inhalt des Verzeichnisses anzuzeigen.
Tippen Sie auf den ”..” an der Spitze der Liste um die Verzeichnisebene zu wechseln.
Die Auswahl wird ignoriert, wenn das aktuelle Verzeichnis bereits das /source /-Verzeichnis ist. Wenn Sie aus Versehen die Load-Taste gedrückt haben, können Sie wieder zurückgehen, indem Sie die BACK-Taste benutzen.

Save
Speichert das derzeitig im Editor befindliche Programm.
Eine Texteingabe-Dialogfeld wird angezeigt. Geben Sie den gewünschten Namen der Datei an, wie diese gespeichert werden soll, und drücken Sie OK. Die Endung. bas wird automatisch hinzugefügt, wenn sie nicht bereits vorhanden ist. Wenn

das aktuelle Programm bereits einen Namen hat, weil es vorher geladen oder gespeichert wurde, dann wird der Name wird in dem Texteingabebereich bereits angezeigt. Sie können eine Sicherungskopie durch Drücken der Taste speichern.

Clear

Das aktuelle Programm im Editor wird gelöscht. Sie erhalten die Möglichkeit das aktuelle Programm zu speichern, wenn es geändert wurde.

Search

Suche nach Zeichenketten im Programm das bearbeitet wird. Gefunden Zeichenketten-Strings können durch eine andere Zeichenfolge ersetzt werden.
Die Suche- Ansicht zeigt ein Textfenster mit dem Text aus dem Editor, einem Feld für den Suchbegriff und einem Feld für das Ersetzen.
Wenn ein Textblock aktuell im Editor ausgewählt ist, dann wird die Suche dort begonnen werden. Sonst wird Suche immer am Beginn des Textes starten, unabhängig davon, wo der Cursor steht.
Nur wenn mit Next ein Text gefunden und ausgewählt wurde, kann dieser Text dann durch den Inhalt des Replace - Feldes ersetzt werden .
Mit dem Replace All Button können alle gefundenen Stellen im Editor Text geleichzeitig ersetzt werden.
Hinweis: Die Suche ignoriert Gross- oder Kleinbuchstaben. Zum Beispiel wird die Suche nach ”basic” auch ”BASIC” finden. Dies geschieht, weil
BASIC! das gesamte Programm in Kleinbuchstaben umwandelt (mit Ausnahme der Zeichen in Anführungszeichen), wenn das Programm gestartet wird.

Next Button

Starten Sie die Suche nach der Zeichenfolge mit dem Feld: Search For - Suche nach.
Die Suche wird an der aktuellen Cursor-Position gestartet.
Wenn die Zeichenfolge gefunden wurde, dann wird sie in Text-Fenster ausgewählt und angezeigt.
Wenn die Schaltfläche Done - Fertig, an dieser Stelle gedrückt wird, dann wird auch im Editorfenster der gefundene Text ausgewählt und angezeigt.
Wenn die Replace -Taste gedrückt wird, dann wird der ausgewählte Text ersetzt.
Drücken Sie die Next- Taste um eine neue Suche zu starten, damit am Ende die Zeichenkette gewählt oder ersetzt werden kann .
Wird kein passender Text gefunden, dann wird eine ”... nicht gefunden”-Meldung angezeigt. Wenn Sie dann die Schaltfläche Done- Fertig

drücken, kehren Sie in den Editor mit dem Cursor am Ende des Programms zurück. Alternativ können Sie den Suchtext ändern und eine neue Suche starten.
Wenn kein Text gefunden wurde, dann wird die Meldung: ”Nichts gefunden um zu ersetzen” zu sehen sein.

Replace all - Alle ersetzen

Alle Vorkommen der gesuchten Textzeichenfolge werden mit dem Button: Replace all durch den gewünschten Text ersetzt. Das Ersetzen beginnt immer am Anfang des Textes. Das letzte Element welches ersetzt wurde, wird im Textfenster ausgewählt dargestellt. Die Anzahl von Elementen die ersetzt wurden, wird in einer Nachricht angezeigt.

Done - Fertig

Geht zurück in den Editor mit dem geänderten Text. Im Textfenster wird dann der Text ausgewählt sein, der geändert wurde.

Back-Taste

Wenn die Back-Taste gedrückt wird, dann wird der Editor mit dem ursprünglichen Text unverändert gezeigt werden. Alle Änderungen die während der Suche gemacht wurden, können so rückgängig gemacht werden. Die Back- Taste kann so als Undo all - alles rückgängig Taste eingesetzt werden.

More->Delete

Der Delete - Löschen-Befehl wird verwendet, um Dateien und Verzeichnisse zu löschen. Der Befehl sollte verwendet werden für die Pflege von Dateien und Verzeichnissen in BASIC!, kann aber auch verwendet werden, um eine beliebige Datei oder ganze Verzeichnisse z.B. auf der SD Card zu löschen.
Drücken Sie auf die Delete- Löschen Taste, dann wird Ihnen der ”BASIC! Datei löschen”-Bildschirm präsentiert. Der Bildschirm zeigt eine sortierte Liste von Dateien und Verzeichnissen. An die Verzeichnisnamen wurde ein (d) angehängt und sie erscheinen am Anfang der Liste.
Tippen Sie einen Dateinamen an, bewirkt das, dass das ”Löschen bestätigen” Dialogfeld angezeigt wird. Drücken Sie die Delete-Taste, um die Datei endgültig zu löschen.
Drücken Sie auf die Schaltfläche No, um das Dialogfeld zu schließen und die Datei nicht löschen.
Wenn ein Verzeichnisname angetippt wird, wird der Inhalt des Verzeichnisses angezeigt. Wenn das Verzeichnis leer ist wird das Löschen bestätigen Dialogfeld angezeigt.
Drücken Sie die Entf-Taste, um das Verzeichnis zu löschen. Drücken Sie die Nein -Taste, um zurückzukehren und das Dialogfenster zu schließen ohne das Verzeichnis zu

löschen.
Durch Drücken der ”..” am oberen Rand des Bildschirms bewegen Sie sich eine Verzeichnisebene nach oben. Das Tippen auf ”..” hat keinen Effekt, wenn
Sie sich bereits im Root-Verzeichnis befinden.
Wenn BASIC! das erste Mal nach der Installation gestartet wird oder nach einem exit wieder neu gestartet wird, wird das
/ SDCard / RFO-basic-Verzeichnis angezeigt. Wenn Sie Verzeichnisse durch frühere Delete-Operationen verändert haben, dann wird das
Verzeichnis dargestellt, indem Sie zuletzt waren.
Verlassen Sie die Delete - Löschen Modus, indem Sie die BACK-Taste benutzen.

More - > Format

Das Programm damit im Editor wird formatiert. Die Schlüsselwörter werden in Großbuchstaben umgewandelt. Programmzeilen werden für die Programmstruktur entsprechend eingerückt werden.

More -> Preferences

Schriftgröße

Legt die Schriftgröße (Small, Medium, Large) fest, um sie in den verschiedenen Bildschirmen in BASIC! zu verwenden.

Editor Screen - Screen Colors

Legt das Erscheinungsbild des Editor-Bildschirm fest. Wählen Sie, schwarzer Text auf weißem Hintergrund, weißer Text auf
schwarzen Hintergrund oder weißer Text auf blauem Hintergrund.

Editor Lines

Aktivieren Sie das Kontrollkästchen, wenn die Textzeilen im Editor mit Zeilenlinien dargestellt werden sollen.

Console-Lines

Aktivieren Sie das Kontrollkästchen, wenn die Textzeilen in der Ausgabe-Konsole unterstrichen werden sollte.

Console-Typefaces

Wählen Sie die Schriftart, die auf der Output-Konsole verwendet werden soll.

Screen Orientation

Wählen Sie hier, ob die Sensoren die Ausrichtung der Bildschirme bestimmen oder eine feste Ausrichtung festlegt werden soll, ohne Rücksicht auf die Sensoren.

Hinweis: Die reverse- Orientierung ist anzuwenden auf Android 2.3 oder neuer.

More -> Commands

Der Commands Button zeigt eine Liste der BASIC! Befehle und Funktionen zum schnellen Kopieren entsprechend der Seitenummern im Anhang A dieses Dokuments. Ein A-Taste bewirkt, dass die Befehlsliste nach unten geht, um Befehle, die mit diesem A- Zeichen beginnen zu zeigen. Es gibt kein Scrollen, wenn es keinen Befehl, der mit diesem A Zeichen beginnt, gibt.
Hinweis: Sie können die Bildschirmtastatur mit der BACK-Taste ausblenden. Wenn Sie das tun, werden Sie nicht in der Lage sein die Befehle-Suchfunktion erneut aufzurufen.
Das Tippen auf einen bestimmten Befehl bewirkt, dass dieser Befehl in die Zwischenablage kopiert wird (ohne die Seiten Nummer) und das Programm kehrt zurück zum Editor. Anschließend können Sie den Befehl in das BASIC! Programm durch langes Drücken einfügen.

More -> About

Die About-Button zeigt die BASIC! Web-Seite von BASIC! entsprichend der Version von BASIC! dass Sie verwenden. Stellen Sie sicher, dass Sie eine Verbindung zum Internet haben, bevor diesen Button auswählen.

More -> Exit

Drücken Sie die Home-Taste, während BASIC! läuft, und dadurch BASIC! verlassen, wird es in genau dem gleichen Zustand bleiben, in dem die Home
Taste gedrückt wurde. Wenn ein Programm ausgeführt wurde, wird es auch noch in dem Zustand ausgeführt werden, wenn BASIC! neu gestartet wird. Wenn Sie
in dem Modus der Löschung waren, wird der Löschen-Bildschirm angezeigt, wenn Basic! neu gestartet wird.
Die einzige Möglichkeit, Basic! sauber zu beenden ist es, den Button exit - Beenden zu verwenden.

Run

Drücken Sie die Menü-Schaltfläche Run - Ausführen und es startet das Programm.
Die BASIC! Output-Konsole wird, sobald das Programm zu laufen beginnt, geöffnet. Sie werden die einzelnen Schritte der Programmausführung nicht alle auf diesem Bildschirm sehen, es sei denn das Programm zeigt (z.B. print-Befehle) etwas oder die END-Anweisung wurde schon ausgeführt, oder Sie arbeiten im Echo-Modus oder gibt es einen Laufzeitfehler. Wenn das Programm nichts druckt (print) oder anzeigt, dann ist der einzige Hinweis den Sie bekommen, wenn das Programm beendet wurde und wenn das Programm mit einer End-Anweisung endet.

Wenn das Programm enthält keine ausführbaren Anweisungen enthält, dann wird die Meldung: ”Nothing to execute” angezeigt.
Das Drücken der BACK-Taste stoppt ein laufendes Programm. Das Drücken der BACK-Taste, wenn das Programm lief, startet den Editor. Wenn das Programm mit einem Laufzeitfehler endet, wird die Linie, wo der Fehler aufgetreten ist im Editor gezeigt.
Wenn der Fehler in einer Include-Datei aufgetreten ist, dann wird die INCLUDE-Anweisung gezeigt und ausgewählt.
Das Scrolling im Editor wird nicht funktionieren solange noch dieser Text ausgewählt ist, Deaktivieren Sie die Textstelle, um wieder scrollen zu können.
Der Editor Cursor bleibt dort, wo er war, als das Programm gestartet wurde, wenn kein Laufzeit-Fehler aufgetreten ist.

Menu
Wenn Sie Menü drücken, während ein Programm läuft oder nachdem das Programm beendet wurde, wird bewirkt, dass das Run-Menü angezeigt wird, außer wenn das Programm im Grafikmodus läuft. (Siehe Pkt. Grafik für Details.)

Stop
Wenn ein Programm läuft, wird das Stop-Menü aktiviert werden. Die Taste Stop stoppt den Lauf des Programms. Stopp wird nicht aktiviert, wenn kein Programm läuft.

Editor
Der Editor wird nicht aktiviert, wenn ein Programm ausgeführt wird. Wenn das Programm angehalten wird, kann der Editor aktiviert werden und wenn Sie dann Editor wählen, kann mit dem Editor erneut eingegeben werden. Sie können auch die BACK-Taste nutzen, um dies zu tun.

Crashes - Abstürze
BASIC! ist ein sehr großes und komplexes Programm, das sich permanent in der Entwicklung befindet. Von Zeit zu Zeit sind Abstürze nicht völlig zu vermeiden. Wenn so ein Unfall passiert, sehen Sie eine kurze Meldung auf dem Bildschirm ”BASIC! Crashed”. Es öffnet sich das Statusfeld, um Bericht zu erstatten, mit der Anzeige ” BASIC! Crashed”. Bitte öffnen Sie das Statusfeld, um das Problem zu melden.
Wenn Sie unten auf dem Statusfeld ziehen, sehen Sie den Eintrag ”BASIC! has crashed”. Bitte klicken Sie hier um das Problem zu melden. Wenn man auf dem Element klickt, wird ein Dialogfeld geöffnet. An dieser Stelle können Sie dem Entwickler einige zusätzliche Informationen über das, was Sie gerade getan haben, liefern und dadurch zum Absturz führte. Sie können auch Ihre E-Mail-Adresse angeben, wenn Sie möchten, dass der Entwickler, mit Ihnen Kontakt zum Problem aufnimmt. Sie können auch auf

Abbrechen drücken und nicht über das Problem berichten. Der Bericht schickt einen Stack-Trace und andere nicht-personenbezogene Informationen, die dem Entwickler helfen das Problem zu beheben. Solche Berichte sind unerlässlich, um aus BASIC! ein noch stabileres Produkt zu machen.

Beschreibung der Syntax der Befehle

Groß- und Kleinbuchstaben

Die Befehle können sowohl in Kleinschreibung oder zur besseren Lesbarkeit mit Gross- und Kleinschreibung geschrieben werden. Es spielt keine Rolle, ob Sie Groß-oder Kleinschreibung verwenden. BASIC! konvertiert jedes Zeichen (mit Ausnahme der in Anführungszeichen) in Kleinbuchstaben, wenn das Programm ausgeführt wird.

<nexp> und <sexp> und <lexp>

Diese Begriffe bezeichnen einen numerischen Ausdruck (‹nexp›) oder einen String-Ausdruck ‹sexp› oder eine logische ‹sexp›
expression - Ausdruck ‹lexp›. Der Ausdruck kann eine Variable, eine Zahl oder eine Zeichenfolge in Anführungszeichen sein
expressions (a*xˆ2 + bx _ c).

<var> and <var$>

Diese Schreibweise wird verwendet, wenn eine Variable, nicht ein Ausdruck, in dem Befehl verwendet werden muss. Arrays mit
Indizes (n [1,2], s $ [3,4]) werden ähnlich als Zeichenketten durch $ oder ohne $ als Zahlen gekennzeichent, verwendet, wie bei ‹var› und ‹var$›.

Array[] and Array$[]

Diese Schreibweise bedeutet, dass ein Array-Name ohne Indizes verwendet werden muss.

{something}

zeigt irgend etwas optional frei wählbares an.
{ A I B I C } Zum Beispiel:
Text.open {r I w I a}, fn....
zeigt die logische Auswahl ODER an, also ”r” oder ”w” oder ”a” muss gewählt werden:
Text.open r, fn ..
Text.open w, fn..
Text.open a, fn..

X ...,X
weist auf eine variable Größe einer Liste von Dingen an, die durch Komma getrennt sind. Mindestens ein Element ist erforderlich.

{,n ...,n}
Gibt eine optionale Liste von Dingen an, die aus null oder mehr Dingen besteht, die durch Komma getrennt sind.

<statement> - Anweisung
Geben Sie hier eine ausführbare BASIC! Aussage an. Ein ‹statement› - Anweisung ist in der Regel eine Codezeile, die innerhalb von Befehlen auftreten kann
wie zum Beispiel: If ‹lexp› then ‹statement›

Numbers - Zahlen
Die Zahlen in BASIC! sind mit doppelter Genauigkeit 64-Bit IEEE 754 Floating Point. Dies bedeutet:

- die angezeigte Zahl besteht immer aus Dezimalpunkt (kein deutsches Komma) und 0. Zum Beispiel: 99 wird angezeigt als ”99.0”. Sie können drucken Zahlen ohne Kommastellen anzeigen oder drucken indem Sie den Format-Befehl. Zum Beispiel Format (”# #”, 99) wird als ”99” gedruckt, verwenden.

- Eine Zahl mit mehr als 7 signifikanten Ziffern wird im Fließkomma-Format ausgedruckt werden. Zum Beispiel, die Zahl 12345678 wird als 1.2345678E7 gedruckt werden. Der Format-Befehl kann auch verwendet werden um große Zahlen in anderem als dem Fließkomma-Format anzuzeigen..
- Mathematische Operationen mit Dezimalwerten sind ungenau. Wenn Sie mit Währungen arbeiten, sollten Sie sollte die Zahl mit 100 multiplizieren, um zu rechnen, um erst nach der Zurückverwandlung (Teilen durch 100) das Ergebnis ausreichend genau anzeigen oder ausdrucken zu können. Sie müssen Dezimalzahlen mit führender Null eingeben. Mit der Eingabe von .15 entsteht ein Syntaxfehler. Mit Hilfe von 0.15 wird kein Syntaxfehler erzeugt..
- Zahlen können in Strings mit dem Format-Befehl oder mit der Str$ (‹nexp›) Funktion umgewandelt werden. Für die Zwecke dieser Dokumentation nennt man Zahlen, die in einem BASIC! Programm erscheinen ”Numerische Constants”.

Strings

Strings -Zeichenketten in BASIC! beginnen und enden mit dem Anführungszeichen (") Zeichen. Zum Beispiel: "Dies ist ein String" ist ein String.

Strings können auch wiederum Anführungszeichen beinhalten, wie zum Beispiel:

```
Print "His name is \"Jimbo\" Jim Giudice".
```

Die Ausgabe ist dann: Sein Name ist "Jimbo" Jim Giudice.

Ein "Neue Zeile"- Zeichen \n kann auch in einen String eingefügt werden

print "Jim \n Giudice"

gibt aus:

Jim

Giudice

Andere Sonderzeichen können in einen String mit der CHR $ ()-Funktion eingefügt werden. Strings mit numerischen Zeichen können auf BASIC! in Zahlen mit dem Val (‹sexp›)-Befehl umgewandelt werden.

Für Zwecke dieser Dokumentation nennt man Zeichenketten, die innerhalb eines BASIC! Programms erscheinen, String Constants.

Variablen

Variablennamen

Eine BASIC! Variable ist ein Container für numerische oder String-Werte. Variablennamen müssen mit den Zeichen "a" bis "z" oder "#" oder "@" beginnen. Die restlichen Zeichen in der Variablen-Namen können die Ziffern 0 bis 9 beinhalten.

Ein Variablenname kann so lang werden wie notwendig.

Großbuchstaben können in Variablennamen verwendet werden, aber es muss bedacht werden, dass sie während der Laufzeit des Programms zu Kleinbuchstaben umgewandelt werden

So ist der Name der Variablen "gLoP" ist der gleiche wie der Name "glop" in BASIC! .

BASIC! Schlüsselwörter sollten nicht verwendet werden, um damit den Namen einer Variablen zu beginnen. Zum Beispiel:

würde Donut = 5 als Do Nut = 5 interpretiert werden. BASIC! wird dadurch erwarten, dass einer Do-Anweisung zu folgen ist, bis irgendwo ein Statement gefunden wird und das Programm endet.

Typen von Variablen

Es gibt zwei Arten von Variablen: Variablen, die Zahlen und Variablen, die Zeichenketten beinhalten. Variablen die Zeichenketten beinhalten, müssen mit dem Zeichen "$" enden.

Variablen die Zahlen beinhalten, enden nicht mit ”$” .
”Alter”, ”Betrag” und ”Höhe” sind alles numerische Variablennamen.
Vorname$, ”Strasse$” und ”A$” sind alles String-Variablen-Namen.

Scalar und Array-Variablen

Es gibt zwei Klassen von Variablen: Scalars und Arrays. Eine scalare Variable kann einen und nur einen Wert beinhalten.
Ein Array-Variable kann viele Werte beinhalten.

Arrays

Ein Array ist so variabel, dass viele Werte in einer systematisch geordneten Art und Weise organisiert gehalten werden können. Die einfachste Art eines Array ist das lineare Array. Es kann auch als eine Kolonne oder als eindimensionale Liste von Werten bezeichnet werden. Das Array A[Index] ist eine lineare Anordnung. Es kann Werte beinhalten, auf die als A zugegriffen werden kann, wie A[1], A [2], .., A [n]. Die Zahl in den eckigen Klammern ist der Index.
Wenn Sie eine Liste von zehn Tieren erhalten wollten, könnten Sie ein Array namens Tiere$ [] erstellen, das einen Zugriff mit einem Index von 1 bis 10 hat. Zum Beispiel ist Tiere$ [5] = ”Katze”
Arrays können mehr als ein Index oder Dimension haben. Ein Array mit zwei Dimensionen können wie ein Tabellenblatt oder eine Liste aus eindimensionalen Listen (Datenkolonnen) beschrieben werden und bestehen aus mehreren Zeilen und Spalten. Gehen wir davon aus, dass wir der Liste von Tieren jeweils drei Merkmale zu jedem Tier zuweisen wollen.
Wie z. B. dem Eintrag für ”Katze” könnte ”schnurrt” und ”hat vier Beine” und ”hat Fell” zugeordnet werden. Wir richten das so ein, dass das Array zwei Dimensionen hat, so dass Tiere$ [5,2] = ”hat vier Beine” beinhalten kann. Wenn jemand fragte, was die Eigenschaften der Katze sind, würde man in Tiere$ [index] suchen, bis ”Katze” wird bei Index = 5 gefunden ist. Index = 5 kann dann verwendet werden um Zugang zu den Eigenschaften Tiere$[Index [{1 | 2 | 3}] zu bekommen.

BASIC! Arrays können eine beliebige Anzahl von Dimensionen haben. Die einzige Einschränkung ist, dass die gesamte Anzahl der Elemente in einem einzelnen Array maximal 10.000 beträgt.
BASIC! Arrays sind nicht auf 0 sondern auf ”Einsen” basiert. Dies bedeutet, dass das erste Element eines Arrays einer Liste einen Index von ”1” hat.
Der Versuch, auf ein Array mit einem Index von ”0” (oder weniger als 0) zu zugreifen

erzeugt einen Laufzeitfehler. Bevor ein Array verwendet werden kann, muss es dimensioniert mit dem DIM-Befehl dimensioniert werden.Der DIM-Befehl weist BASIC! an, wie viele Indizes verwendet werden sollen und die Anzahl der Elemente pro Index. Einige BASIC! Befehle bestimmen automatisch die Dimensionen eines Arrays. Details zu automatisch dimensionierten Arrays werden in der Beschreibung für diese Befehle dargestellt.
Hinweis: Es wird empfohlen, an Stelle von eindimensionalen Arrays die list - Befehle zu verwenden. Die list - Befehle bieten mehr Flexibilität als die Array-Befehle.

Array-Befehle

Diese Befehle können auf alle Arrays in einer oder der anderen Weise angewendet werden.

Dim Array [<nexp>. . . , <nexp>]

Der DIM-Befehl weist BASIC! an wie viele Dimensionen ein Array haben und wie groß diese Dimensionen sind. Mehrere Arrays können mit einer Dim-Anweisung dimensioniert werden. String-und numerische Arrays können mit einem einzigen Befehl dimensioniert werden. Beispiele sind:

```
DIM A[15]
DIM B$[2,6,8], C[3,1,7,3], D[8]
```

UnDim Array[]

Zum Un-Ent-Dimensionieren eines Arrays kann mit diesem Befehl das Array wieder entdimensioniert werden und anschließend mit neuen unterschiedlichen Dimensionen belegt werden. Der Befehl ist sehr nützlich, wenn in einer Schleife unter Verwendung von Befehlen, die automatische Dimensionierung eines Array angewendet wird. Das Array [] wird dann ohne Index angegeben. Der Befehl ist genau das gleiche wie ”array.delete”

Array.average <Average_nvar>, array []

Sucht den Durchschnitt aller Werte in einem numerisches Array Array [], und dann lädt das Ergebnis in ‹Average_nvar›. Das Array [] wird ohne Index angegeben.

Array.Copy sourceArray [{{<start>, <length>}], DestinationArray [{{-} <extras>}]

Das zuvor dimensionierte oder geladende (array.load) sourceArray kann auf das neue DestinationArray Gebiet kopiert werden.
Die Kopie wird mit dem ‹Start› Element des sourceArray begonnen, wenn der optionale Parameter ‹start› angegeben ist. Wenn ‹start› 0 oder 1 ist oder nicht vorhanden ist, dann wird die Kopie mit dem ersten Element des SourceArray beginnen.

Der optionale Parameter ‹length› gibt eine bestimmte Anzahl von Elementen an, die aus dem sourceArray kopiert werden.
Wenn ‹length› nicht ‹start› + ‹length› die Anzahl der Elemente in dem gegenwärtigen sourceArray übersteigt, dann wird das gesamte Array aus ‹start› bis zum Ende des Arrays kopiert werden. Der optionale Parameter ‹extra› gibt an, dass die ‹extra› Anzahl leere Elemente, dem Zielarray vor oder nach der Kopie hinzugefügt werden sollen.
Die zusätzlichen, leere Elemente werden an den Anfang des Arrays angefügt,
, wenn das optionale Minuszeichen (-)-Zeichen vorhanden ist. Wenn minus nicht vorhanden ist, dann werden die zusätzlichen, leeren Elemente so zugegeben, dass sie die Anordnung im Array beenden.
Die Anordnung kann entweder numerische oder String-Arrays beinhalten, aber sie müssen beide vom gleichen Typ sein. Die ‹extra› Elemente für ein numerisches Array werden auf Null gesetzt. Die zusätzlichen Elemente für ein String-Array sind ein leerer String "".
Sehen Sie sich die Datei Beispielprogramm, f26_array_copy.bas, für die Arbeit mit diesem Befehl als Beispiel an.

Array.delete Array[]
Macht das Gleiche wie UnDim Array [].

Array.length <Length_nvar>, array []
Reserviert oder bestimmt die Anzahl der Elemente im Array [] in ‹Length_nvar›.

Array.load Array [], {<nexp>, <nexp> .., <nexp>}
Lädt das Array [] mit der Liste, ‹nexp› {, ‹nexp› .., ‹nexp›}, von Werten. Das Array [] kann nur eine einzige Dimension als Liste von Werten haben.
Das Array muss jedoch nicht vorher dimensioniert werden.
Das Array wird ohne einen Index angegeben.
String-Arrays können in der gleichen Weise geladen werden, wenn die numeric expression ‹nexp› durch ‹sexp› ersetzt werden.
Die Liste der string expressions ‹sexp› kann durch Beendung der Zeile mit dem Zeichen " ~ " dann in der nächsten Zeile fortgesetzt werden.
Beispiele sind:
Array.load Numbers[], 2, 4, 8 , n^2, 32
Array.load Hours[], 3, 4,7,0, 99, 3, 66~
37, 66, 43, 83~
83, n*5, q/2 +j
Array.load Letters$[], a, b,c,d$,e

Array.max <Max_nvar> Array []
Sucht das Maximum aller Werte in numerisches Array, Array [], und lädt dann das Ergebnis in ‹Max_nvar›. Array [] wird ohne Index angegeben.

Array.min <Min_nvar>, array []
Sucht das Minimum aller Werte in numerisches Array Array [], und dann lädt das Ergebnis in ‹Min_nvar›. Array [] wird ohne Index angegeben.

Array.variance <v_nvar>, array []
Findet die Varianz aller Werte in numerisches Array Array [], und lädt das Ergebnis in ‹v_nvar›. Array [] wird ohne Index angegeben.

Array.reverse Array [] | $ Array []
Kehrt die Reihenfolge der Werte in dem angegebenen Array um. Array [] wird ohne Index angegeben.

Array.shuffle Array [] | $ Array []
Mischt die Werte des angegebenen Arrays in zufälliger Reihenfolge. Array [] wird ohne Index angegeben.

Array.sort Array [] | $ Array []
Sortiert die Werte des angegebenen Arrays in aufsteigender Reihenfolge. Array [] wird ohne Index angegeben.

Array.std_dev <sd_nvar>, array []
Findet die Standardabweichung aller Werte im numerischen Array Array [], und lädt das Ergebnis in ‹sd_nvar›. Array [] wird ohne Index angegeben.

Array.sum <Sum_nvar>, array []
Ermittelt die Summe aller Werte in numerisches Array Array [], und legt dann das Ergebnis in die ‹Sum_nvar› Variable. Array [] wird ohne Index angegeben.

Datenstrukturen und Zeiger - Pointer in BASIC!
BASIC! bietet Befehle, welche die Arbeit mit Data Strukturen erleichtern und die mit traditionellen Basic- Softwarevarianten bisher nicht möglich waren. Diese Kommandos bieten die Einfügung von Listen, Bundles, Stacks und Queues an.
Das zentrale Konzept hinter der Umsetzung dieser Befehle (und vieler anderer BASIC! Befehle) ist der Zeiger - Pointer. Ein Zeiger ist ein numerischer Wert, ein Index in einer Liste oder einer Tabelle von Dingen.
Als ein Beispiel für die Arbeit mit Zeigern, kann man sich einem Aktenschrank mit Ordnern vorstellen, in den Sie Einträge machen. Der Aktenschrank als Kabinett aus

Ordnern und Dateien wird betreut durch Ihre Assistentin. Sie sehen nie die Datei selbst, in die Sie eintragen. Im Rahmen Ihrer Tätigkeit werden Sie einen neuen Ordner anlegen in welchen Sie einiges Informationsmaterial geben. Sie geben dann den Ordner Ihrer Assistentin um ihn an einen zugewiesenen Platz im Regal zu stellen. Die Assistentin schreibt eine eindeutige Nummer auf den Ordner und gibt Ihnen einen Zettel mit dieser eindeutigen Nummer. Sie können später diesen Ordner abgerufen, indem Sie Ihrer Mitarbeiterin bitten, Ihnen den Ordner mit der betreffenden Nummer drauf zu bringen.
In BASIC! erzeugen Sie ein Informations-Objekt (Ordner). Dann setzen Sie dieses Informations-Objekt in BASIC! in eine virtuelle Schublade. Von BASIC! erhalten Sie nun eine eindeutige Nummer (Pointer) für dieses Informations-Objekt.
In der Folgezeit verwenden Sie diese Nummer (Zeiger- Pointer), um besondere Informationen aus dem Objekt abzurufen.
Weiter mit der Ordner Analogie. Nehmen wir an, dass Sie Ordner haben, die Informationen enthalten über Kunden. Diese Informationen könnten Dinge wie Name, Adresse und Telefonnummer sein. Die Zahl, die Ihre Assistentin beim Füllen des Ordnern an den Kunden vergibt, wird die Kunden-Nummer genannt werden. Sie können Informationen zu einem Kunden abrufen, indem sie die Assistentin bitten, Ihnen den Ordner zu bringen der die betreffende Kundennummer beinhaltet. In BASIC! würden Sie ein Objekt mit Kundeninformationen als Bundle (Bündel, Menge) verwenden, um die Kundeninformationen als Objekt zu erstellen.
Der Zeiger (der Zettel) welcher BASIC! Ihnen zurückgibt, wenn Sie die Kundendaten zusammenstellen oder bündeln und so ein Bündel (Bundle) erstellen, wird die Kundennummer sein.
Nun wollen wir davon ausgehen, dass ein Kunde etwas bestellt. Sie brauchen das Bundle, das alle Kundendaten enthält, die für die Bestellung nötig sind. Solche unterschiedlichen Bündel werden durch die Abteilung Auftragsabwicklung, für die Rechnungserstellung verwendet und vielleicht sogar in der Marketing-Abteilung (um den Kunden für ähnliche Produkte zu werben). Bündel dort könnten z.B. bestellte Artikel, den Preis, usw. enthalten. Das Bündel müsste eigentlich aber auch Informationen über den Kunden enthalten. Anstatt noch einmal zusätzlich, die Kunden-Informationen doppelt anzulegen, erstellen Sie einfach ein Kunden-Nummer Feld, das die Kunden-Nummer (Pointer) enthält. Der Zeiger, bewirkt dass Sie die gewünschten Kundendaten bekommen, wenn sie die Bestellung bearbeiten. Das neue Bundle erhält die Bestellnummer. Sie können verschiedene Listen und Bündel für den Einsatz in verschiedenen Abteilungen erstellen. Nun wäre es schön, eine Liste aller Aufträge

herzustellen, die ein Kunde aus dem Kundenbündel gemacht hat. Sie würden dazu eine Liste aller Nummern von Bestellungen von diesem Kunden erstellen. Beim Erstellen dieses neuen Kundenbündels, würden Sie ihre Assistentin - BASIC! beauftragen eine leere Liste erstellen. BASIC! würde ihnen einen Merkzettel oder Zeiger auf dieses leere Liste zurückgeben. Sie würden dann diese Merkzettel/Zeiger in die Kundenakten legen. Später, wenn ein Kunde eine Bestellung aufgibt, würden Sie mit diesen Merkzettel/Zeiger die leere Liste abrufen und holen lassen und dann die Bestellnummer auf der leeren Liste hinzufügen.
Möglicherweise möchten Sie auch andere Listen von Bestellungen als Bündel/Bundles für andere Zwecke erstellen. Sie können beispielsweise eine Liste mit ausgeführten Aufträgen füllen, eine andere Liste mit nicht ausgeführten Aufträgen und noch eine anderer Liste mit stornierten Aufträgen usw.
Alle diese Listen würden einfach nur Listen mit Bestellnummern sein. Jede Bestellnummer würde auf das Bestellnummer- Bündel und das wiederum auf das Kunden- Bündel hinweisen.
Wenn Sie das tatsächlich schaffen, haben sie eine Datenbank in BASIC! erstellt und es kann sein, Sie möchten nun alle diese Pakete aus Bündeln auf externen Speichermedien sichern.
Wie Sie nun diese Informationen aus den internen Datenstrukturen auf externe Speichermedien bekommen, das ist im Moment eine anspruchsvolle Benutzeraktion.

Lists - Listen

Eine List - Liste ist vergleichbar mit einem eindimensionalen Array. Der Unterschied liegt in der Art und Weise wie eine Liste gebaut und eingesetzt wird. Ein Array muss vor der Verwendung dimensioniert werden. Die Anzahl der Elemente, die in dem Array platziert werden sollen, müssen vorher bestimmt werden. Eine Liste dagegen startet leer und wächst nach Bedarf. Sie wächst wie es erforderlich ist und Elemente können entfernt, ersetzt und überall in die Liste eingefügt werden.
Ein weiterer wichtiger Unterschied ist, dass eine Liste nicht auf einen Variablentyp festgelegt ist. Ein numerischer Zeiger wird zurückgegeben, wenn eine Liste erstellt wird. Alle weitere Zugänge zu der Liste erfolgen mit Hilfe dieses numerischen Zeigers. Dadurch ist ist es sehr einfach, eine Liste von Listen zu machen. Eine Liste von Listen ist nichts anderes, als eine numerische Liste mit numerischen Zeigern oder Verweisen auf andere Listen.
Listen können in neue Arrays kopiert werden. Arrays können zu Listen hinzugefügt werden.
Alle diese List- Befehle werden mit der Sample-Programmdatei, f27_list.bas

demonstriert.

List Commands List-Befehle

List.create N | S, <pointer_nvar>

Erstellt eine neue, leere Liste deren Typ durch den N-oder S-Parameter spezifiziert ist. Ein String-Typ-Liste wird erstellt, wenn der Parameter ”S” angegeben wurde. Ein numerischer Typ Liste wird erstellt, wenn der ”N”- Parameter angegeben wurde.
Der Pointer/Zeiger/Merkzettel/Verweis/Link auf die neue Liste wird mit der numerischen ‹pointer_nvar› Variable zurückgegeben.
Die neu erstellte Liste ist leer. Die Größe, die für eine neu erstellte Liste zurückgegeben wird, ist Null.

List.add <pointer_nexp>, <nexp>{,<nexp>..,<nexp>}

Die Werte, ‹nexp› {, ‹nexp›.., ‹nexp›} wird zum angegebenen Liste hinzugefügt.
Strings können in der gleichen Weise hinzugefügt werden, wenn dass ‹nexp› ersetzt wird, durch ‹sexp›. Die Liste der ‹sexp› kann auch durch Beendigung der Zeile mit dem Zeichen ”~” in der nächsten Zeile fortgesetzt werden.
Beispiele sind:

```
List.add Nlist, 2, 4, 8 , n^2, 32

List.add Hours, 3, 4,7,0, 99, 3, 66~
37, 66, 43, 83~
83, n*5, q/2 +j

List.add Name ”Bill”, ”Jones”~
”James”, ”Barnes”~
”Jill”, ” Hanson”
```

List.add.list <destination_list_pointer_nexp>, <source_list_pointer_nexp>

Die Elemente in der Quell-Liste werden ans Ende der Ziel-Liste hinzugefügt werden.
Die beiden Listen müssen vom gleichen Typ sein.

List.add.array <destination_list_pointer_nexp>, Array $ [] | array []

Die Elemente des angegebenen Arrays werden zum Ende der Ziel-Liste hinzugefügt werden.
Der Array-Typ müssen die gleichen wie in der Liste Typ sein.
Das Array wird ohne einen Index angegeben.

List.replace <pointer_nexp>, <index_nexp>, <sexp> | <nexp>

Die List-Elemente gekennzeichnet durch ‹index_nexp›, auf die der Zeiger ‹pointer_nexp› zeigt, werden durch
den String oder numerischen Ausdruck ersetzt.
Die Index ist auf die Zahl 1 bezogen. Das erste Element der Liste ist 1.
Den Typ des Ersatz-Ausdrucks (numerisch oder String) muss mit dem ursprünglich erstellten Listentyp übereinstimmen.

List.insert <pointer_nexp>, <index_nexp>, <sexp> | <nexp>

Der ‹sexp› oder ‹nexp› Wert wird in die Liste, auf die der pointer-Zeiger zeigt, eingefügt werden.
Das Element wird an dem gegebenen Index eingefügt.
Der Index ist auf die Zahl 1 bezogen. Das erste Element der Liste ist 1.
Das eingefügte Element des Ausdruckstypes muss zu dem Typ (String oder numerisch) der verwendeten Ursprungsliste passen.

List.remove <pointer_nexp>, <index_nexp>

Das List-Element, was durch ‹index_nexp› spezifiziert ist, wir aus der Liste mit Zeigernummer ‹pointer_nexp› entfernt.
Der Index ist auf die Zahl 1 bezogen. Das erste Element der Liste ist 1.

List.get <pointer_nexp>, <index_nexp>, <svar> | <nvar>

Die List-Element, das durch ‹index_nexp› spezifiziert ist, wird aus der Liste mit Zeigernummer ‹pointer_nexp› in die angegebene Zeichenfolge oder numerische Variable zurückgegeben.
Der Index ist auf die Zahl 1 bezogen. Das erste Element der Liste ist 1.
Das der Variablentyp des Rückgabeelements muss zu dem Typ (String oder numerisch) der verwendeten Ursprungstabelle passen.

List.type <pointer_nexp>, <svar>

Die Art der Liste mit der Zeigernummer ‹pointer_nexp› wird als String-Variable zurückgegeben.
Der Großbuchstabe ”S” wird zurückgegeben, wenn die Liste ist eine Liste von Strings ist.
Der Großbuchstabe ”N” wird zurückgegeben, wenn die Liste ist eine numerische Liste von Zahlen ist.
The size of theList pointed to by the List pointer will be returned in the numeric variable.

List.size <pointer_nexp>, <nvar>

Die Größe der Liste, mit der entsprechenden Zeigernummer wird in der numerischen Variable zurückgeben.

List.clear <pointer_nexp>

Die Liste, mit der entsprechenden Zeigernummer wird gelöscht. Die Größe der Liste dadurch gleich Null sein.

List.search <pointer_nexp>, value|value$, <result_nvar> , {,<start_nexp>}

In der Liste, mit der entspechenden Zeigernummer wird die angegebene Zeichenfolge oder ein numerischer Wert gesucht. Die Position an der das Gesuchte in der Liste gefunden wurde, wird als numerischen Variablen zurückgegeben. Wenn der Wert nicht gefunden wurde, wird die zurückgegebene numerischen Variable den Wert Null haben. Wenn der optionale Startausdruck-Parameter vorhanden ist, beginnt die Suche an dem angegebenen Element. Der Standardwert ist 1.

List.ToArray <pointer_nexp>, Array$[] | Array[]

Die Liste, mit der entsprechenden Zeigernummer wird in das neue, bisher nicht dimensionierte Array kopiert werden. Die angegebene Array-Typ (String oder numerisch) müssen vom gleichen Typ wie der, der Liste sein.

Bundles

Ein Bündel/Bundle ist eine Gruppe von Werten, die zu oder in einem einzigen Objekt gesammelt wurden. Zu einem Bündel-Objekt kann eine Anzahl von Strings und numerischen Werten zusammengebunden werden.
Die Werte werden gesetzt und Zugang erhält man mit Schlüsseln. Ein Schlüssel ist ein String, der den Wert identifiziert. Zum Beispiel kann ein Bündel eine Person mit Vornamen und Nachnamen enthalten. Die Schlüssel für den Zugriff auf diese Zeichenfolgen könnten sein:
”vorname” und ”nachname”. Ein numerischer Wert fürs Alter könnte im Bundle z.B. mit dem ”alter”- Schlüssel belegt werden
Ein neues, leeres Bündel wird mit dem Befehl bundle.create erstellt. Der Befehl erstellt einen numerischen Zeiger auf das leere Bündel. Die Tatsache, dass das Bündel nur durch einen numerischen Zeiger dargestellt wird, bedeutet, dass
die Bündel auch in Listen gesetzt werden können. Bundles können auch in anderen Bundles enthalten sein. Das heißt, die Kombination aus Listen und Bündeln kann verwendet werden, um beliebig komplexe Datenstrukturen zu erstellen.
Nachdem ein Bundle erstellt wurde, können Schlüssel und Werte auf das Bündel mit dem bundle.put Befehl hinzugefügt werden.
Diese Werte können über die Schlüssel mit dem bundle.get Befehl abgerufen werden.
Es gibt noch andere Bundle- Befehle, um die Verwendung von Bündeln zu erleichtern.

Bundle Commands

Bundle.create <pointer_nvar>

Ein neues, leeres Bundle erstellt wird. Die Bundle Zeigernummer wird in ‹pointer_nvar› zurückgegeben.

Beispiel:

Bundle.create bpter

Bundle.put <pointer_nexp>, <key_sexp>, <value_nexp> | <value_sexp>

Der Wert ‹value_nexp› oder ‹value_sexp› wird in dem angegebenen Bundle, unter dem angegebenen Schlüssel ‹key_sexp› platziert werden.

Der Typ des Wertes wird durch den Typ des Ausdruckswertes bestimmt.

Beispiel:

Bundle.put bptr, ”vorname”, ”Frank”

Bundle.put bpter,”alter”, 44

Bundle.get <pointer_nexp>, <key_sexp>, <nvar> | <svar>

Der Wert der durch den Schlüssel-Ausdruck bestimmt ist, wird durch numerische oder String-Variablen zurückgegeben. Die (String oder numerische) des Ziel-Variable wird in dem Variablentyp des Schlüssel gespeichert.

Beispiel:

Bundle.get bpter,”vorname”, vorname$

Bundle.get bpter,”alter”, alter

Bundle.keys <pointer_nexp>, <list_nvar>

Eine Liste der aktuellen Schlüssel in dem genannten Bündel werden in eine neue Liste, mit deren zurückgegebenen Zeiger ‹list_nvar› platziert.

Die Schlüssel-Namen in der zurückgegebenen Liste, können extrahiert mit den verschiedenen List-Befehlen benutzt werden.

Beispiel:

```
bundle.keys bpter, list
list.size list, size
for i = 1 to size
 list.get list, i, key$
 bundle.type bpter, key$, type$
 if type$ = "S"
            bundle.get bpter, key$, value$
            print key$, value$
       else
            bundle.get bpter, key$, value
            print key$, value
            endif
```

```
next i
```

Bundle.contain <pointer_nexp>, <key_sexp>, <contains_nvar>

Wenn der Schlüssel im angegebenen Schlüssel-String-Ausdruck in dem Bündel enthalten ist, dann gibt die ”contain-enthält”- numerische Variable einem Wert ungleich Null zurück. Der zurückgegebene Wert ist null, wenn der Schlüssel nicht enthalten ist.

Bundle.type <pointer_nvar>, <key_sexp>, <type_svar>

Gibt den Typ (String oder Numerisch) des angegebenen Schlüssel in der angegebenen String-Variablen zurück. Die Variable ‹typr_svar› enthält dann den Großbuchstabe ”N”, wenn der Schlüssel numerisch ist. Die ‹typr_svar› enthält einen Großbuchstaben ”S”, wenn der Typ des Schlüssels, ein String ist.

Beispiel:

```
Bundle.type bpter, ”alter”, type$
Print type$ (wird ausgeben N)
```

Bundle.clear <pointer_nvar>

Im Bundle, auf das der Pointer ‹pointer_nvar› zeigt, werden aller Verknüpfungen gelöscht. Dadurch erhält man ein leeres Bündel aus Stacks -Stapel.

Stacks (Stapelverarbeitung)

Stacks (Stapel) sind wie ein Magazin für eine Pistole.

Die letzte Kugel die ins Magazin gesteckt wurde, ist die erste Kugel aus dem Magazin. Dies gilt auch, in etwa vergleichbar für Stacks. Der letzte Objekt das in dem Stapel abgelegt ist, ist die erste Aufgabe aus dem Stapel. Dies wird als LIFO (Last In First Out) (Letzter der kam, geht als Erster Raus) bezeichnet.

Ein Beispiel für die Verwendung eines Stapels ist der BASIC! GOSUB Befehl. Wenn ein GOSUB- Befehl ausgeführt wird, wird die Zeilenkennzeichnung, für die Rückkehr auf einem Stapel abgelegt. Wenn eine Rückkehr ausgeführt wird, wird die Rücklauf-Zeilenkennzeichnung aus dem Stapel geholt. Diese Methodik erlaubt es, dass GOSUB-Befehle auf allen Ebenen verschachtelt werden. Jede Rückkehr wird immer auf die Zeilenkennzeichnung zurück erfolgen, aus der das letzte GOSUB ausgeführt wurde.

Ein ausführbares Beispiel für Stapelverarbeitung oder Stacks kann im Beispielprogramm Datei, f29_stack.bas gefunden werden.

Stack-Befehle

Stack.create N|S, <ptr_nvar>
Erstellt einen neuen Stapel der bezeichneten Art (N = Anzahl. S = String). Der Stack-Pointer in ‹ptr_nvar›.

Stack.push <ptr_nexep>, <nexp>|<sexp>
Schiebt die ‹nexp› oder ‹sexp› Variable auf den Stapel, der mit ‹prt_nexp› bezeichnet ist. Die Art der Variable, die geschoben wird, muss zur Art des erzeugten stacks-Stapels passen.

Stack.pop <ptr_nexep>, <nvar>|<svar>
Wirft den obersten top-of-stack-Wert aus dem mit ‹ptr_nexep› bezeichneten Stapel aus und legt ihn in die Variable ‹nvar› oder ‹svar›.
Der Typ des Wertes muss mit dem Typ des erstellten Stapel zusammenpassen.

Stack.peek <ptr_nexep>, <nvar>|<svar>
Gibt den top-of-stack-Wert des Stapels ‹ptr_nexep› in die mit ‹nvar› oder ‹svar› bezeichnete Variable. Der Wert wird auf der Spitze des Stapels erhalten bleiben.
Der Typ des variablen Wertes muss mit dem Typ des erstellten Stapels zusammenpassen.

Stack.type <ptr_nexep>, <svar>
Der Typ (Zeichenfolge oder Zahl) des Stapels, der mit ‹ptr_nexep› bezeichnet wird, wird als ‹svar› zurückgegeben. Wenn der Stack numerisch ist, wird der Großbuchstaben ”N” zurückgegeben. Wenn der Stapel eine Zeichenfolge ist, wird das Zeichen ”S” zurückgegeben.

Stack.IsEmpty <ptr_nexep>, <nvar>
Wenn der Stapel ‹ptr_nexep› leer ist, wird der Wert 1 in ‹nvar› zurückgegeben. Wenn der Stapel nicht leer ist,wird der Wert 0 sein.

Stack.clear <ptr_nexep>
Der Stapel, mit ‹ptr_nexep› bezeichnet, wird gelöscht.

Queues - Warteschlangen
Eine Queue - Warteschlage ist wie die Warteschlage, die sich bei Ihrer Bank bildet. Wenn Sie ankommen, erhalten Sie den letzten Platz in der Reihe oder an der Warteschlange. Wenn ein Bankangestellter fertig wird, wird die Person an der Spitze der Reihe oder an der Warteschlange entfernt, die durch diesen Kassierer bedient wurde. Die ganze Reihe bewegt sich um eine Person nach vorne. Eventuell sind auch Sie irgendwann der Kopf der Reihe und Sie werden vom verfügbaren Kassierer betreut

werden. Eine Warteschlange ist so etwas ähnliches wie ein Stapel, aber eben mit der Verarbeitung First In First Out (FIFO) (Erster der kam, geht auch als Erster raus) statt Stapel (LIFO) (Letzter der kam, geht als Erster raus) angewendet wird.
Mit unserem Kundenauftragsbearbeitung- Analogie, könnten Sie eine Warteschlange als eine Ordnung für Arbeitsbündel für die Auftrags- Bearbeitungsabteilung ansehen. Neue Arbeitsbündel würden an das Ende der Warteschlange gesetzt werden. Das an der Sitze stehende (top an the queu- bundle) Arbeitsauftragsbündel würden durch die Abteilung Auftragsabwicklung entfernt werden, sobald die Dienstleistung erledigt wurde und man bereit für eine neue Auftragsbearbeitung ist.
Es gibt keine speziellen Befehle in BASIC! für Queue-Operationen. Wenn Sie eine Warteschlange erstellen möchten, erstellen Sie eine Liste - List.
Verwenden Sie list.add, um neue Elemente an das Ende der Warteschlange hinzuzufügen.
Verwenden Sie list.get um das Element am Anfang der Warteschlange abzurufen und verwenden Sie list.remove, um dass Element von der Spitze der Warteschlange zu entfernen Sie sollten natürlich auch list.size benutzen, bevor Sie list.get aufrufen, um sicherzustellen, dass in der Warteschlange überhaupt ein Element verblieben ist.

Comments - Kommentare

! - Einzeiliger Kommentar

Wenn das erste Zeichen in einer Zeile ist das ”!” Ausrufezeichen ist, betrachtet BASIC! die gesamte Zeile als einen Kommentar und ignoriert den Inhalt.
Wenn das ”!” an anderer Stelle in der Zeile erscheint, wird das Folgende nicht als Kommentar angesehen.

!! - Block-Kommentar

Wenn eine Zeile mit dem ”!!” doppelten Ausrufezeichen beginnt, werden alle Zeilen die folgen als Kommentare angesehen und werden von BASIC! ignoriert. Der Kommentarblock- Abschnitt endet an der nächsten Zeile, die mit ”!!” gekennzeichnet ist.

% - Kommentar In der Mitte der Zeile

Wenn das Zeichen ”%” in einer Zeile erscheint (außer in einem String in Anführungszeichen), dann ist der Rest der Zeile ein Kommentar.

Expressions - Ausdrücke

Numeric <nexp> := {<numeric variable>|<numeric constant}
{<noperator>
<nexp>|<end of line>}

Numeric Operators <noperator>

Die numerischen Operatoren werden in Rangfolge aufgelistet. Vorrangige Operatoren werden ausgeführt, bevor Operatoren mit niedrigere Priorität ausgeführt werden. Der Vorrang kann durch Klammern geändert werden.

Unary +, Unary (Vorzeichen alleinstehend)

Exponent ^

Multiplizieren *, / Dividieren

Addieren +, Subtrahieren -

Numerische Beispiele für Ausdrücke

a

a*b + 4/d 2*(d^2)

a + b + d + rnd (0)

b + ceil(d/25) + 5

Pre- und Post-Inkrement-Operatoren

++ X

Erhöht (inkrementiert) den Wert von x um 1 bevor der Wert X verwendet wird

--y

Reduziert (dekrementiert) den Wert von y um 1 bevor der Wert y verwendet wird

x++

Inkrementiert den Wert von x um 1, nachdem der X-Wert verwendet wurde

y--

Dekrementiert den Wert von y um 1 nachdem der y-Wert verwendet wurde.

Hinweis: Diese Vorgänge werden durch ein Präprozessor implementiert, welcher den Quellcode ändert, bevor er ausgeführt wird. Haben Sie einen Syntaxfehler in einer Zeile, die diese Operatoren enthält, wird die Codezeile anders aussehen.

Op vergleichbare aufgabenbezogene Vorgänge

+=

a+= 1 ist das gleiche wie a = a + 1

*=

b* = 5 + 3 ist das gleiche wie b = b * (5+3)

-=

c -= d ist das gleiche wie c = c - d

/=

e/ = log(37) + 1 ist das gleiche wie e = e / (log(37) + 1)

Hinweis: diese Vorgänge werden durch ein Präprozessor implementiert, der den Quellcode ändert, bevor er ausgeführt wird. Haben Sie einen Syntaxfehler in einer Zeile, die diese Operatoren enthält, wird die Codezeile anders aussehen.

String <sexp> := {<string variable>|<string constant>} { + <sexp> | <end of line>}

Es gibt nur einen String-Operator: +
Dieser Operator wird als Verkettungsoperator eingesetzt. Es wird verwendet, um zwei String zu einer Zeichenkette zu verbinden
Print abc + def
Ausgabe ist: abcdef

Logical <lexp>

Logische Ausdrücke erzeugen Falsch (false) oder Wahr (true) Ergebnisse. False und True sind in BASIC! vertreten durch den numerischen Wert Null und nicht Null. False = 0 ist. True = nicht 0.
Es gibt zwei Arten von logischen Ausdrücken: Numerische logische Ausdrücke und Zeichenketten-String- logische Ausdrücke.
Beide Arten produzieren ein numerisch dargestelltes wahr oder falsch.

‹slexp› := {‹string variable› I ‹string constant›} ‹logical operator›{‹string variable› I ‹string constant›}

‹nlexp› := {‹numeric variable› I ‹ numeric constant›} ‹logical operator›{‹ numeric variable› I ‹ numeric constant›}

Es gibt auch den vorzeichenähnlichen (unären) Nicht- NOT (!) Operator. NOT invertiert den Wahrheitswert eines logischen Ausdrucks.

Logische Operatoren

Die logischen Operatoren sind entsprechend Vorrang mit der höchsten Priorität zuerst aufgeführt. Die Rangfolge kann durch Klammern geändert werden.

Unary Not !

Weniger als ”‹”, Größer als ”›”, Kleiner oder gleich ”‹=”, Größer als oder gleich ”› =”

Gleichheitszeichen ”=”, nicht gleich ”‹›”

und (and) ”&” oder (or) ”|”

Beispiele für logische Ausdrücke

1 ‹ 2 (true)

3 ‹› 4 (true)

a ‹ bcd (true)

!(1 & 0) (true)

Zuweisungsoperationen

Variablen erhalten ihre Werte durch Zuweisungen. Zuweisungsanweisungen sind von der Form:

‹nvar› = ‹nexp›

‹svar› = ‹sexp›

Die ursprünglichen Basic-Sprache verwendet den Befehl, LET, um eine Zuordnung zu erreichen:

LET ‹nvar› = ‹nexp›

BASIC! hat auch den Befehl LET aber er ist optional. Das einzige Mal, wo Sie LET verwenden könnten wäre, wenn Sie einen Variablennamen haben wollen, der mit einem BASIC! Schlüsselwort beginnt.

Zum Beispiel:

Letter$ = ”B” wird von Basic! grundsätzlicher gesehen als

LET ter$ = ”B”

Wenn Sie wirklich Letter $ als Variable verwenden wollen, können Sie es bedenkenlos verwenden, indem Sie eine LET-Anweisung verwenden:

LET Letter$=B

Wenn Sie die Zuweisung in einer IF-Anweisung verwenden, sollten Sie auch den Befehl LET benutzen:

If 1 ‹ 2 then LET letter$=B

Op vergleichbare aufgabenbezogene Operationen

+=

a+= 1 ist das gleiche wie a = a + 1

*=

b* = 5 + 3 ist das gleiche wie b = b * (5+3)

-=

c -= d ist das gleiche wie c = c - d

/=

e/ = log(37) + 1 ist das gleiche wie e = e / (log(37) + 1)

Hinweis: diese Vorgänge werden durch ein Präprozessor implementiert, der den Quellcode ändert, bevor er ausgeführt wird. Haben Sie einen Syntaxfehler in einer Zeile, die diese Operatoren enthält, wird die Codezeile anders aussehen.

Mathematische Funktionen

Mathematische Funktionen verhalten sich wie eine numerische Variable ‹nexp› (oder ‹lexp›).

BOR(<nexp1>, <nexp2>)

Der logische bitweise Wert von ‹nexp1› ODER ‹nexp2› wird zurückgegeben. Die Fließkomma-Doubles werden
vor der Operation auf ganze Zahlen umgerechnet.
BOR(1,2) = 3

BAND(<nexp1>, <nexp2>)

Die logische bitweise Wert von ‹nexp1› UND ‹nexp2› wird zurückgegeben. Die Fließkomma-Doubles werden
vor der Operation auf ganze Zahlen umgerechnet.
BAND (3,1) = 2

BXOR (<nexp1>, <nexp2>)

Die logische bitweise XOR-Wert von ‹nexp1› ‹nexp2› wird zurückgegeben. Die Fließkomma-Doubles werden
vor der Operation auf ganze Zahlen umgerechnet.
BXOR (7,1) = 6

ABS (<nexp>)

Gibt den absoluten Wert von ‹nexp›zurück.

SQR (<nexp>)
Gibt die korrekt gerundete positive Quadratwurzel aus ‹nexp› zurück.

CBRT (<nexp>)
Gibt die Kubikwurzel aus ‹nexp›.

RANDOMIZER(<nexp>)
Erzeugt eine zufälligen Nummern Generator für die Benutzung mit der RND() Funktion. Die Randomizer() Function gibt immer Null zurück. Wenn die numerische Erweiterung = 0 ist, dann wird ein Generator der die Tageszeit als Zufallsbasis nutzt, erzeugt. Wenn dieser ‹› 0 ist, dann wird ein Generator erzeugt, der diesen numerischen Erweiterungswert als Basis benutzt. Zufällige Zahlen die mit einer Nicht-Null Basis erzeugt wurden, werden wieder immer die gleichen Zahlenfolgen mit der Zufallzahlerzeugung RND (‹nexp›) erzeugen. Die generierten Zufallszahlen werden größer als Null und kleiner als 1 sein.

(0 ‹ = n ‹ 1)

RND ()
Erzeugt eine Zufallszahl durch den Zufallszahlgenerator. Wenn ein Zufallsgenerator (Randomizer) nicht zuvor ausgeführt wurde, wird dann ein neuer Zufallsgenerator erstellt mithilfe von ”Randomizer(0)”.

CEIL (<nexp>)
Rundet auf. 3.X wird 4 und -3.X wird -3.

FLOOR (<nexp>)
Rundet ab. 3.X wird 3 und -4 wird zu -3.X .

MOD (<nexp1>, <nexp2>)
Gibt den Rest der ‹nexp1› durch ‹nexp2› geteilt zurück.

ROUND (<nexp>)
Rundet auf oder ab und liefert die nächste ganze Zahl zu ‹nexp›.

LOG(<nexp>)
Gibt den natürlichen Logarithmus (Basis e) von ‹nexp›.

LOG10(<nexp>)
Gibt den Basis 10 Logarithmus von ‹nexp›.

EXP (<nexp>)
Ergibt die e hoch ‹nexp› Potenz.

POW (<nexp1>, <nexp2>)
Ergibt ‹nexp1› potenziert mit ‹nexp2›..

SIN (<nexp>)
Ergibt den trigonometrischen Sinus des Winkels ‹nexp›.

COS (<nexp>)
Ergibt den trigonometrischen Kosinus des Winkels ‹nexp›.

TAN (<nexp>)
Ergibt den trigonometrischen Tangens des Winkels ‹nexp›.

COSH (<nexp>)
Liefert den trigonometrischen hyperbolischen Kosinus des Winkels ‹nexp›.

SINH (<nexp>)
Ergibt den trigonometrischen hyperbolischen Sinus des Winkels ‹nexp›.

HYPOT (<nexp_x>, < Nexp_y)
Sqrt(x2+y2) ohne zwischengeschalteten über- oder Unterlauf gibt.

ATAN2 (<nexp_x>, <nexp_y>)
Gibt den Winkel Theta aus der Umwandlung der rechtwinkligen Koordinaten (x, y) in Polarkoordinaten (r, Theta) zurück.

TODEGREES (<nexp>)
Konvertiert den ‹nexp› Winkel im Bogenmaß zu einer annähernd gleichwertigen Winkel in Grad gemessen.

TORADIANS (<nexp>)
Konvertiert den ‹nexp› Winkel gemessen in Grad in einen annähernd gleichwertigen Winkel in Bogenmaß.

ASIN (<nexp>)
Liefert den Arcus Sinus des Winkels, ‹nexp›, im Bereich von-pi / 2 bis pi / 2.

ACOS (‹nexp›)

Liefert den Arcus Cosinus des Winkels, ‹nexp›, im Bereich von 0,0 bis pi.

ATAN (<nexp>)
Ergibt den Arcustangens des Winkels ‹nexp›, im Bereich von-pi / 2 bis pi / 2.

VAL(<sexp>)
Konvertiert die ‹sexp› in eine numerische Zahl.

LEN (<sexp>)
Ergibt die Länge der Zeichenkette ‹sexp› an.

HEX (<sexp>)
Konvertiert die String-Ausdruck, der eine Hexadezimalzahl darstellt, in eine Dezimalzahl.

Oktober (<sexp>)
Konvertiert den String-Ausdruck, der eine Oktalzahl darstellt in eine Dezimalzahl.

BIN (<sexp>)
Konvertiert die String-Ausdruck, der eine binäre Zahl darstellt, in eine Dezimalzahl.

SHIFT (<value_nexp>, <bits_nexp>)
Bit-Verschiebung des Wert-Ausdrucks um die angegebene Anzahl von Bits. Wenn der Bit-Ausdruck ‹0 ist, wird der Wert nach links verschoben. Wenn der Bit- Ausdruck› 0, werden die Bits nach rechts verschoben. Die rechte Shift wird die Replikation des Vorzeichenbit.

Clock ()
Ergibt die Zeit in Millisekunden seit dem letzten Systemstart an.

ASCII(<sexp>)
Ergibt den ASCII-Wert des ersten Zeichens von ‹sexp›. Wenn ‹sexp› eine leere Zeichenfolge (””) ist wird der Wert 256 zurückgegeben.

Is_In (<Search_for_sexp>, <Search_in_sexp> {, <start_nexp>}
Sucht in der Zeichenfolge ‹Search_in_sexp› den ‹Search_for_sexp› String.
Wenn die optionale ‹start_nexp› vorhanden ist, dann wird die Suche bei diesem Wert gestartet, sonst wird die Suche bei 1 beginnen. Das Zeichen ‹start_nexp› muss größer oder gleich 1 sein. Wenn die {Search_for_sexp› nicht in ‹Search_in_sexp› gefunden wurde, ist der zurückgegebene Wert 0, sonst entspricht der zurückgegebene Wert dem Index, an dem ‹Search_in_sexp› in der Zeichenkette gefunden wurde.

Starts_with (<Search_for_sexp>, <Search_in_sexp> {, <start_nexp>}
Wenn die Suche nach ‹Search_for_sexp› ab ‹start_nexp› in (‹Search_for_sexp› beginnt, dann muß mindestens noch die Länge des
‹Search_for_sexp› vorhanden sein, sonst wird Null zurückgegeben.
Wenn der optionale ‹start_nexp› vorhanden ist, dann wird die Suche bei diesem Wert starten, sonst wird die Suche
bei 1 mit dem ersten Zeichen derZeichenkette beginnen. Daraus folgt ‹start_nexp› muss› = 1 sein.

Ends_with (<look_for_sexp>, <look_in_sexp>)

Wenn die ‹look_in_sexp› nicht mit ‹look_for_sexp› enden, dann wird der zurückgegebene Wert Null sein. Wenn der
Ausdruck am Ende den angegebenen Ausdruck hat, wird der zurückgegebene Wert der Index sein, wo der String
‹Look_for_sexp› beginnt. Der Wert wird immer größer oder gleich 1 sein.

gr_collision (<object_1_nvar>, <object_2_navr>)

Das Objekt ‹nvars› wird eine Objekttabellen Nummer haben, wenn die Objekte erstellt wurden. Wenn sich die Grenzen der Boxen der beiden Objekte überlappen, dann wird die Funktion true (nicht Null) zurückgegeben. Wenn sie das nicht tun und
nicht überlappen, dann wird die Funktion false (Null) zurückgegeben.
Objekte, die auf Kollision geprüft werden können, sind: Rechteck, Bitmap-, Kreis, Bogen und Oval. Im Falle eines Kreis, Bogen oder eines ovales Objektes, kommt nur die rechteckige Begrenzungs Box, für die Kollisionsprüfung zum Einsatz,
nicht das eigentliche gezeichnete Objekt.

Background ()

Wenn ein laufendes BASIC! Programm weiterläuft, wenn die Home-Taste gedrückt wird, verursacht das einen Lauf im Hintergrund. Wenn BASIC! nicht Hintergrund-Modus, läuft, steht BASIC! im Vordergrund Modus. BASIC! verlässt den Hintergrund-Modus und tritt in den Vordergrund Modus, wenn das BASIC! Symbol auf dem Home-Bildschirm angetippt wurde. Manchmal will ein BASIC! Programmierer will wissen, ob ein Programm im Hintergrund läuft. Ein Grund dafür könnte sein, weil Musik aufhört zu spielen, während es im Hintergrund-Modus läuft.
Die background()-Funktion gibt true (1), wenn das Programm im Hintergrund läuft. Sie gibt false (0), wenn das Programm nicht im Hintergrund läuft.
Wenn Sie in der Lage sein wollen, den Hintergrund-Modus zu erkennen, während sie im Graphik Modus sind, müssen Sie gr.render nicht aufrufen, während Sie im Hintergrund-Modus arbeiten. Wenn Sie das tun kann das Programm nicht mehr ausgeführt werden, bis der Vordergrund Modus neu eingegeben wird. Nutzen Sie folgende Code Zeile für alle gr.render Befehle:
If !background() then gr.render

String-Funktionen

GETERROR$()

Ergibt eine Fehlermeldung, die von ”OnError” abgefangen aber nicht ausgegeben

wurde.

CHR $ (<nexp>)

Gibt das Zeichen des verwendeten Zeichensatzes, das dem Wert des ‹nexp› entspricht, zurück.
Das Zeichen ”C” ist hexadezimal 43,
Print chr$(16*4 + 3) Ergibt die Anzeige: C
‹nexp› kann Werte größer als 255 haben und kann somit verwendet werden, um Unicode-Zeichen zu generieren.

Left $ (<sexp>, <nexp>)

Ergibt die am weitesten links stehende ‹nexp› Anzahl Zeichen von ‹sexp›.
Wenn ‹nexp› = 0, dann wird eine leere Zeichenfolge (””) zurückgegeben.
Wenn ‹nexp› ist größer ist als die Länge des ‹sexp› dann wird nur der gesamte String zurückgegeben.

MID$ (<sexp>, <start_nexp>, <Count_nexp>})

Ergibt eine Zeichenzeichenkette, bestehend aus ‹Count_nexp› Anzahl Zeichen, beginnend bei ‹start_nexp›.
Ein Null-Wert für ‹start_nexp› wird zu 1 geändert werden.
Wenn ‹start_nexp› ist größer als die Länge des ‹sexp› dann wird eine leere Zeichenfolge (””) zurückgegeben.
Wenn ‹Count_nexp› größer als die Länge des ‹sexp› werden nur die Zeichen ‹Start_nexp› bis zum Ende der ‹sexp› zurückgegeben.
‹Count_nexp› ist optional. Wenn der Parameter nicht ‹Count_nexp› vorhanden ist, dann werden die restlichen Zeichen aus ‹Start_nexp› bis zum Ende der Zeichenfolge, zurückgegeben.

REPLACE$ (<target_sexp>, <argument_sexp>, <replace_sexp>)

Ersetzt ‹target_sexp› mit allen Instanzen des ‹argument_sexp› durch ‹replace_sexp›.

RIGHT $ (<sexp>, <nexp>)

Liefert einen String, aus der am weitesten rechts stehenden ‹nexp› Anzahl Zeichen, der Zeichenkette ‹sexp›.
Wenn ‹nexp› = 0, dann wird eine leere Zeichenfolge (””) zurückgegeben.
Wenn ‹nexp› größer ist als die Länge des ‹sexp› dann wird nur der gesamte String zurückgegeben.

STR$ (<nexp>)
Wandelt einen numerischen Wert in eine Zeichenkette.

LOWER$ (<sexp>)
Liefert ‹sexp› allein in Kleinbuchstaben.

UPPER $ (<sexp>)
Liefert ‹nexp› allein Großbuchstaben.

VERSION$()
Gibt die Versionsnummer von BASIC! als String zurück.

HEX$(<nexp>)
Ergibt einen String in hexadecimaler Darstellung des numerischen Ausdrucks.

OCT$(<nexp>)
Ergibt einen String in octaler Darstellung des numerischen Ausdrucks.

BIN$(<nexp>)
Ergibt einen String in binärer Darstellung des numerischen Ausdrucks.

Format$ (<Pattern_sexp>, <nexp>)
Ergibt einen String für den Wert ‹nexp› der durch das Muster ‹Pattern_sexp› formatiert wird. Das erste Zeichen von FORMAT generiert wird ein Minus (-) Zeichen für Zahlen ‹ 0 oder ein Leerzeichen für Zahlen › = 0.

Leading Sign - (Führendes Zeichen/Initiale)
Das erste Zeichen von Format erzeugt ein Minus (-) Zeichen für Zahlen ‹0 oder ein Leerzeichen für Zahlen› = 0 .
Wenn die Zahl› = 0, dann wird das erste Zeichen immer ein Leerzeichen sein.

Floating Field
Wenn das erste Zeichen des Musters ‹pattern_sexp› nicht "#" oder "." oder "!" ist, dann wird aus dem Zeichen (zusammen mit dem führenden Zeichen/Initiale) ein Floating-Feld. Für diesen Mustertyp ist das Zeichen $ typisch. Wenn kein floating Zeichen angeboten wird, dann wird es durch ein Leerzeichen ersetzt.
Der Floating-Bereich wird immer zwei Zeichen breit sein.

Decimal Point- Komma
Ein Muster kann nur ein und allein nur ein optionales Dezimalzeichen haben. Wenn das Dezimalzeichen nicht in dem Muster verwendet wird, dann werden keine

Nachkommastellen ausgegeben. Die Zahl der ”#”-Zeichen nach dem Dezimalzeichen, bestimmt die Anzahl der Dezimalstellen, die ausgegeben werden.

Pattern-Zeichen

Jedes Zeichen ”#” wird durch eine Ziffer aus der Zahl in der Ausgabe ersetzt. Wenn es mehr ”#”-Zeichen als Anzahl Ziffern vorhanden sind, dann werden die # - Zeichen durch Leerzeichen ersetzt.

Musterzeichen%

Jedes ”%” Zeichen wird durch eine Ziffer aus der Zahl in der Ausgabe ersetzt. Wenn es mehr ”%”-Zeichen als Anzahl Ziffern in der Zahl vorhanden sind dann wird die Anzahl Ziffern% durch die das Null (”0”) Zeichen ersetzt.

Überlauf

Wenn die Anzahl der Stellen, die Anzahl der # und %-Zeichen überschreitet, dann werden in die Ausgabe die Zeichen ** anstatt des floating field eingefügt.
Am besten ist es, die # und %-Zeichen nicht zu mischen, da dies zu unerwarteten Ergebnissen führen kann.

Nicht Muster Zeichen

Wenn jedes Zeichen in dem Muster (außer dem ersten Zeichen) kein # oder % -Zeichen ist, dann werden die Zeichen
direkt in der Ausgabe kopiert. Diese Möglichkeit wird in der Regel für Kommas verwendet.

Output Size - Ausgabe Länge

Die Anzahl der ausgegebenen Zeichen entspricht immer der Anzahl der Zeichen in dem Muster plus zwei floating Zeichen.

Beispiele:

Format$(##,###,###, 1234567) Ausgabe: 1,234,567
Format$(%%,%%%,%%%.#, 1234567.89) Ausgabe: 01,234,567.8
Format$($###,###, 1234) Ausgabe: $1,234
Format$($###,###, -1234) Ausgabe: -$1,234

Benutzerdefinierte Funktionen (User- Funktion)

Benutzerdefinierte Funktionen sind BASIC! Funktionen ähnlich wie abs (n), mod (a, b) und left$ (a$,n), aber diesmal werden die Regeln für die Funktion durch den Benutzer definiert. Benutzer-Funktionen sollten in der Regel am Anfang des Programms definiert werden und insbesondere im Vorfeld der Orte, an denen sie aufgerufen werden.

Jedesmal, wenn ein User- Funktion wiederum aus einer User- Funktion aufgerufen wird, wird eine bestimmte Menge an Speicher für den Ausführungsstapel verwendet. Die Tiefe dieser verschachtelten Aufrufen wird deshalb durch die Größe des Speichers begrenzt, dass Ihr besonderes Android-Gerät der Anwendung zuweist.

Befehle

Fn.def Name | Name $ ({nvar} | {} svar | array [] | $ Array [], .. {nvar} | {} svar | array [] | $ Array [])

Wenn der Name der Funktion mit dem $-Zeichen endet wird die Funktion einen String zurückgeben, sonst wird sie eine Nummer zurückgeben. Die Liste der Parameter kann so viele Parameter wie nötig haben, von keinem bis zu Dutzenden. Die Parameter können numerisch oder string, scalar oder array sein.
Im Folgenden sind alle gültig:

```
fn.def pi()
fn.def cut$(a$, left, right)
fn.def sum(a, b, b, d, e, f, g, h, i, j)
fn.def sort(v$[], direction)
```

Es gibt zwei Arten von Parametern: call by reference und call by value. Call-by-value bedeutet, dass der Aufruf- Variablenwert (oder Ausdruck) in die Aufruf- Variable kopiert werden. Änderungen an der genannten Variable innerhalb der Funktion wirken sich nicht auf den Wert der Aufruf- Variable aus. Call by reference bedeutet, dass der aufrufende Variablenwert geändert wird, wenn der aufgerufene variable Wert innerhalb der Funktion geändert wird.
Scalar (Nicht-Array)-Funktions Variablen können entweder ein call by value oder ein call by value-Typ sein. Welchen Type die Variable will, ist abhängig davon wie sie aufgerufen wurde. Wenn die Aufruf- Variable das Zeichen ”&” am Anfang hat, dann wird die Variable der call by reference Typ sein. Wenn es kein ”&” vor dem Aufruf-Variablennamen hat, dann wird die Variable ein call by value-Typ sein.
Beispiel:

```
Fn.def test(a)
a=9
fn.rtn a
fn.end
a =1

print test(a), a % Ausgabe: 9, 1
print test(&a), a % Ausgabe: 9, 9
```

Array-Parameter sind immer ein call by reference Typ.
Beispiel:

```
Fn.def test(a[])
a[1] = 9
fn.rtn a[1]
fn.end
dim a[1]
a[1] = 1

print test(a[]), a[1] %Ausgabe: 9, 9
```

Die Funktionen können andere Funktionen aufrufen. Eine Funktion kann auch rekursiv sich selbst aufrufen. Alle Variablen innerhalb einer Instantiierung einer Funktion sind privat reserviert für diese Instantiierung der Funktion. Eine Variable im Hauptprogramm namens v$ ist nicht die gleiche Variable v$ innerhalb einer privaten Funktion. Ferner ist ein Variable v$ in einer rekursiv aufgerufenen Funktion mit dem Namen v$ nicht diesselbe v$ wie in der aufrufenden Funktion.

Fn.rtn <sexp> | <nexp>

Bewirkt die Funktionsausführung zu beenden. Der Wert in ‹sexp› I ‹nexp› wird der Rückgabewert der
Funktion sein. Die Rückgabe- Ausdruckstyp, String oder Zahl, muss mit dem Typ des Namen der Funktion übereinstimmen.
fn.rtn Aussagen können sich an beliebiger Stelle im Programm, wo sie gebraucht werden befinden.

Fn.end

Dieser Befehl beendet die Definition einer benutzerdefinierten Funktion. Jede Definition einer Funktion muß enden mit
fn.end. Wenn die fn.end Anweisung ausgeführt wurde, wird ein Standardwert zurückgegeben. Wenn die Funktion
numerisch ist, dann wird der Wert 0.0 zurückgegeben. Wenn die Funktion ist eine Zeichenfolge, wird ein leerer String
(””) zurückgegeben.

Call <user_defined_function>

Führt die definierte Benutzerfunktion aus. Ein bisheriger Wert der von der Funktion schon

zurückgegeben wurde, wird verworfen. Verwendet eine Benutzer Funktion als Parameter für einen Benutzer-Funktionsaufruf.
Der folgende Code:
CALL fun1 (fun2(3), fun3$(”cat”), 3)
erzeugt einen Laufzeitfehler, trotz der Tatsache, dass es syntaktisch korrekt ist. Dies entsteht aufgrund eines Fehlers im BASIC!.
Dieses Problem kann auf diese Weise umgangen werden:
CALL fun1 (fun2 (3) + 0, fun3$(”cat”) + ””, 3)

Program Control-Befehle

If - Else -Elseif- Endif

Die IF- (Wenn-dann) Anweisungen sorgt für die Ausführung von Anweisungen nach der Erfüllung von Bedingungen. Die Formen die zur Verfügung stehen sind:
IF ‹Bedingung› {THEN}
‹Anweisung›
‹Anweisung›
.
‹Anweisung›
{ ELSEIF‹Bedingung› { THEN }
‹Anweisung›
‹Anweisung›
..
‹Anweisung›}
{ ELSE
‹Anweisung›
‹Anweisung›
‹Anweisung›}
ENDIF

-Oder-
If ‹Bedingung› THEN ‹Anweisung› {ELSE ‹Anweisung› }

Sehen Sie sich die Datei Beispielprogramm, F04_if_else.bas, für das Arbeiten mit dem IF-Befehl an.

For - To - Step - Next

For ‹nvar› = ‹nexp_1› To ‹nexp_2› {Step ‹nexp_3›}
‹Anweisung›
..
‹Anweisung›
Next {‹nvar›}
{Schritt ‹nexp_3›} ist optional und kann weggelassen werden. Wenn die Schrittweite weggelassen wird, ist der Schritt- Wert 1.
‹nvar› wird dem Wert von ‹nexp_1› zugeordnet und
‹nvar› wird mit ‹nexp_2› verglichen.
Wenn ‹nexp_3› positiv ist und ‹nvar› ‹= ‹nexp_2› dann
werden die Anweisungen zwischen dem For und Next ausgeführt.
Auch wenn ‹nexp_3› negativ ist und dann ‹nvar› › = ‹nexp_2› dann werden
die Anweisungen zwischen dem For und Next ausgeführt.
Wenn die nächste Anweisung ausgeführt wird, wird ‹nvar› um 1 oder die Schrittweite erhöht oder reduziert und der Test der Bedigungen kann wiederholt werden. Die eine oder mehrere ‹Anweisung›en werden dann so oft ausgeführt werden, bis der Test der Bedingungen wahr ist.
For-Next- Schleifen können auf jeder Ebene verschachtelt werden. Wenn For-Next-Schleifen verschachtelt sind, wird immer die nächste innere Next-Anweisung ausgeführt werden. Das bleibt so, egal mit welchem ‹nvar› das Next bezeichnet ist. Aus praktischen Gründen sollte das Next mit einem ‹nvar› bezeichnet sein, es stellt aber nicht mehr als einen Kommentar dar.

F_n.break

The For-Next loop will be terminated if this statement is executed within an active For-Next loop. The statement immediatelyfollowing the Next will be executed.

While <lexp> - Repeat (Weil <die Bedingung besteht> - erfolgt Wiederholung)

While ‹lexp›
‹Anweisung›
.
‹Anweisung›
Repeat
Die ‹Anweisung› -en zwischen WHILE und REPEAT werden so lange ausgeführt, wie die Bedingung ‹lexp› wahr ist.

Die ‹Anweisung› -en werden überhaupt nicht ausgeführt werden, wenn die Bedingung ‹lexp› bereits false beginnt.
While-Schleifen kann man auf jeder Ebene verschachteln. Wenn While-Schleifen verschachtelt sind, wird die nächste repeat-Anweisung auf die innere nächste While-Schleife angewendet .

W_r.break

Die While / Repeat-Schleife wird beendet, wenn diese Anweisung ausgeführt wird und eine While/ Repeat-Schleife aktiv ist. Die Anweisung wird unmittelbar nach der Wiederholung ausgeführt.

Do - Until<lexp>

Do

‹statement›

‹statement›

Until ‹lexp›

Die Anweisungen zwischen Do und Until werden ausgeführt, bis ‹lexp› wahr wurde. Die ‹statement› -Anweisungen werden immer mindestens einmal ausgeführt.
Do-Schleifen können beliebig verschachtelt werden. Jede aufgetretene UNTIL-Anweisung, gilt für die zuletzt ausgeführten DO- Anweisung.

D_u.break

Die Do / Until-Schleife wird beendet, wenn diese Anweisung ausgeführt wurde und eine Do / Until-Schleife aktiv ist. Die Anweisung wird unmittelbar nach der Wiederholung ausgeführt.

GoSub <label>, Return

Die nächste Anweisung, die ausgeführt werden soll, folgt auf die Markierung ‹label› im Programm. Die Anweisungen von dieser Stelle beginnend werden auch weiter ausgeführt, bis eine return-Anweisung angetroffen wird.
Dann wird Ausführung wieder bei der Anweisung, die auf die GoSub-Anweisung folgt, fortgesetzt.
Beispiel:

```
Message$ = ”Hab einen guten Tag”
GoSub xPrint
```

```
Print ”Danke”
‹Anweisung›
.
‹Anweisung›
END
xPrint:
Print Message$
Return
```

Dieser Programmteil erzeugt die Ausgabe:
Hab einen guten Tag
Danke

GoTo <label>

Die nächste (-en) Anweisung (-en) die ausgeführt werden, wird auf die Markierung ‹label› folgen.
Ein ‹label› ist eine Variable am Beginn einer Zeile, die mit einem Doppelpunkt ”:” Zeichen endet. Solche Markierungen müssen
allein auf einer Zeile stehen.
Zum Beispiel:

```
Loop:
‹Anweisung›
.
‹Anweisung›
GoTo Loop
```

Run <filename_sexp> {, <data_sexp>}

Dieser Befehl beendet den Lauf des aktuellen Programms und läd und startet ein BASIC! Programm mit dem Dateinamen der im Zeichenfolgenausdruck benannt ist. Der Dateiname ist relativ zu ”/sdcard/rfo-basic/source/”. Wenn der Dateiname lautet” program.bas”dann wird die Datei ”program.bas” in ”/sdcard/rfo-basic/source/” ausgeführt.
Der optionale Daten-Zeichenfolgenausdruck sorgt für die Weitergabe von Daten an das nächste Programm. Auf die übergebenen Daten kann in dem nächsten Programm durch Verweis auf die spezielle Variable, ##$ zugegriffen werden.
Run-Programme können aufgereiht arbeiten. Ein Programm wird geladen und mit Hilfe des Ausführen des Run-Befehl können andere Programm-Dateien auch laufen.

Diese Kette kann so lange wie nötig sein. Wenn das letzte Programm in einer Kette endet, wird durch Drücken der Zurück-Taste das ursprüngliche Programms im Display des BASIC! Editor gezeigt.

Switch Commands - Schaltbefehle

Die Schaltbefehle werden verwendet, um verschachtelte if-then-else Operationen an unterschiedliche Ziele oder Fälle umzuschalten.

```
Sw.begin a
Sw.case 1
        ‹statement1›
        ‹statement2›
Sw.break
Sw.case 2
        ‹statement3›
        ‹statement4›
Sw.break
Sw.case 3
        ‹statement5›
        ‹statement6›
Sw.break
Sw.default
        ‹statement7›
Sw.end
```

Betätigung der Sw Switch Umschalt- Operationen

Switch Operationen können nicht verschachtelt werden. Fügen Sie keine Switch-Operationen in anderen Switch-Operationen ein.

Sw.begin <nexp> | <sexp>

Beginnt eine Switch-Betrieb.

Die angegebenen ‹nexp› oder ‹sexp› werden geprüft und verglichen mit den

sw.case statements.

Wenn ‹begin_nexp› = ‹case_nexp› oder wenn ‹begin_sexp› = ‹case_sexp› stimmt, dann wird die Anweisung nach dem

sw.case (Ziel, Fall) ausgeführt.

Sw.case <nexp>|<sexp>

Der Typ des Parameters kann numerisch oder String sein und muß dem Typ des

Ausdrucks aus sw.begin entsprechen. Wenn mehrere sw.case Anweisungen die gleichen Parameter haben, wird nur die erste sw.case mit dem passenden Parameter ausgeführt werden.

Sw.break

Sobald ein passender sw.case gefunden werden dann die Anweisungen die nach dem sw.case folgen ausgeführt bis ein sw.break angetroffen wird.
Wenn ein sw.break angetroffen wird, dann sucht BASIC! nach dem sw.end Kommando. Die Ausführung wird dann wieder mit der Anweisung, die nach der sw.end folgt, fortgesetzt. Wenn kein sw.break in einem bestimmten sw.case vorhanden ist, dann werden nachfolgende sw.cases ausgeführt werden, bis ein sw.break auftritt.

Sw.default

Wenn kein passendes sw.case gefunden wird, wird die sw.default, falls vorhanden, ausgeführt. Die sw.default muß
nach all den sw.case Aussagen plaziert werden

Sw.end

Die sw.end beendet einen Switch-Betrieb. Sw.end muss deshalb immer einem sw.begin folgen.

OnError:

Wenn eine ”OnError:” Markierung in einem BASIC! Programm vorhanden ist, dann wird die Steuerung, die Anweisung nach dieser OnError: Markierung in dem Fall ausführen, wenn ein Laufzeitfehler auftritt. Die Fehlermeldungen die abgefangen wurden werden nicht ausgegeben. Die Fehlermeldung kann durch die getError$()-Funktion abgerufen werden.
Seien Sie vorsichtig. Eine Endlosschleife entsteht, wenn ein Laufzeitfehler innerhalb der OnError-Anweisungen auftritt
Sie sollten nicht eine OnError: Anweisung in Ihr Programm plazieren, bevor das Programm vollständig ausgetestet ist.
Eine vorzeitige Anwendung von OneError: macht Schwierigkeiten das Programm zu debuggen.
Die OnError: Markierung muss alleine auf einer Zeile stehen.

OnBackKey:

Durch Drücken der Zurück-Taste hält man in der Regel die Programmausführung an.
Eine OnBackKey: Markierung empfängt das Zurück
Tasten- Benutzungsereignis und die Steuerung geht zur der Anweisung nach der

OnBackKey: Markierung. Wenn das Programm im Grafikmodus läuft, dann sollte mit dem OnBackKey Code der Lauf beendet werden. Wenn dies nicht der Fall
ist, dann kann es sein, dass das Programm nicht zu beenden ist (außer man benutzt eine Task Killer-Anwendung). Die primäre Absicht dieses Abschnittes ist es, damit die Programm Zustands- und Status-Informationen zu erhalten,
bevor es beendet wird.
Die onBackKey: Markierung muss alleine auf einer Zeile stehen.

Back.Resume

Wenn ein Back-Taste-Drücken von OnBackKey gefangen wurde, bewirkt das back.resume, dass das Programm, an dem Punkt wieder aufzunehmen ist, wo die Back-Taste gedrückt wurde.

OnMenuKey:

Wenn dieser Ausdruck im Programm ist, und der Benutzer die Menü-Taste drückt, wird das aktuell laufende Programm unterbrochen. Die Anweisungen nach dieser Bezeichnung werden ausgeführt.

MenuKey.Resume

Wenn ein Tastendruck der Menütaste von OnMenuKey gefangen wurden hat bewirkt das menu.resume, dass das Programm, an dem Punkt wieder aufgenommen wird, wo die Menü-Taste gedrückt wurde.

OnKeyPress:

Wenn dieser Ausdruck im Programm ist, und der Benutzer irgend eine Taste drückt, wird das aktuell laufende Programm unterbrochen. Die Anweisungen nach dieser Bezeichnung werden ausgeführt.

Key.Resume

Wenn ein Tastendruck von OnKey aufgefangen wurde bewirkt key.resume, dass das Programm, an dem Punkt wieder aufzunehmen ist, wo die Taste gedrückt wurde.

End

Die End-Anweisung gibt END auf dem Textausgabe- Bildschirm aus und stoppt die Ausführung des Programms. Ende
Anwisungen können überall im Programm platziert werden.

Exit

Die Exit-Anweisung bewirkt, den Programmablauf zu stoppen, BASIC! zu beenden und auf den Android-Startbildschirm zurückzukehren.

Debug-Befehle

Mit Debug-Befehlen können Sie Ihr Programm testen. Die Ausführung aller Debug-Befehle wird
gestartet durch den debug.on Befehl. Alle der Debug-Befehle werden ignoriert, es sei denn der debug.on
Befehl wurde bereits ausgeführt. Dies bedeutet, dass Sie alle Ihre Debug-Befehle in Ihrem Programm lassen können und sind sicher, dass sie nur ausgeführt werden, wenn Sie sich mit debug.on auf Fehlersuche machen.

Debug.on

Schaltet den Debug-Modus ein. Alle Debugging-Befehle werden im Debug-Modus ausgeführt.

Debug.off

Schaltet den Debug-Modus aus. Alle Debugging-Befehle (außer debug.on) werden ignoriert.

Debug.echo.on

Schaltet den Echo-Modus an. Im Echo-Modus wird jede Zeile des laufenden BASIC! Programm angezeigt, bevor sie ausgeführt wird. Dies kann eine große Hilfe bei der Fehlersuche sein. Die letzten Zeilen die ausgeführt wurden, sind meist die Ursache für das Programm- Problem. Der Echo-Modus wird entweder durch den Befehl debug.echo.off oder den debug.off ausgeschaltet werden.

Debug.echo.off

Schaltet den Echo-Modus.

Debug.Print

Dieser Befehl entspricht genau dem Befehl print, außer dass die Ausgabe nur während des Debug-Modus erfolgt.

Debug.dump.scalars

Druckt eine Liste aller Skalarvariablen mit Namen und Werten. Skalare Variablen sind die Variablennamen, die keine
Arrays oder Funktionen sind. Unter anderem wird dieser Befehl helfen falsch geschriebene Variablennamen zu erkennen.

Debug.dump.array Array []

Gibt den Inhalt des angegebenen Arrays aus. Wenn das Array mehrdimensional ist, wird das gesamte Array in einer linearen Weise dargestellt.

Debug.dump.bundle <bundlePtr_nexp>
Schreibt einen Bundle, auf den der numerische Bundel- Pointer Ausdruck zeigt .

Debug.dump.list <listPtr_nexp>
Schreibt eine Liste, auf die der numerische Liste Pointer Ausdruck zeigt.

Debug.dump.stack <stackPtr_nexp>
Schreibt einen Stapel, auf die der numerische Stack-Pointer Ausdruck zeigt.

Eingabe und Ausgabebildschirm (Console I / O)

Output-Konsole
BASIC! hat zwei Arten von Ausgabe- Bildschirmen: Der Ausgang per Schrift- Konsole und der Grafik-Bildschirm. Der folgende Abschnitt befasst sich
mit der Schrift Augabe- Konsole. Für Informationen über den Grafikbildschirm, siehe den Abschnitt über Grafiken.
Informationen werden auf dem Bildschirm mit dem Befehl print ausgegeben. Auch BASIC! Laufzeit Fehlermeldungen werden
auf diesem Bildschirm angezeigt.
Es gibt keine frei wählbare Ausgabeplätze auf diesem Bildschirm. Die Zeilen werden immer eine Zeile nach der anderen gedruckt.

CLS
Die CLS-Befehl löscht die bisherige Anzeige der Output-Konsole.
Print ‹sexp› I ‹nexp› {, I ;} . . . . ‹sexp› I ‹nexp›{, I ;}
Wenn das Komma (,) als Trennzeichen verwendet wird, wird nur ein Komma wird zwischen den Ausdrücken angezeigt.
Wenn das Semikolon (;) als Trennzeichen verwendet wird, wird die Ausdrücke nichts trennen.
Wenn das Semikolon am Ende der Zeile verwendet wird, wird das Ergebnis der gesamten Ausgabe erst angezeigt, wenn ein letzter Print- Befehl ohne
Semikolon am Ende, ausgeführt wurde.
Beispiele:
Print ”New”, ”Message”
Ausgabe: Neu, Message
Print ”New”;” Message”
Ausgabe: New Message
Print ”New” + ” Message”
Ausgabe: New Message

Print 100-1; ” Luftballons”
Ausgabe: 99.0 Luftballons
Print format$(##, 99) ; ” Luftballons”
Ausgabe: 99 Luftballons
Print A;B;C;
Ausgabe: keine
Print D;E;F
Ausgabe: ABCDEF

Console.save <filename_sexp>

Der aktuelle Inhalt der Textbildschirm- Konsole wird in der Textdatei mit dem Dateinamen, den der Zeichenfolgenausdruck angibt, am angegebenen Speicherort gespeichert.

User Input

Input <Prompt_sexp>, <nvar>|<svar>, {<Default_sexp>| <Default_nexp>}

Generiert einen Eingabedialog.
Die ‹Prompt_sexp› Zeichenfolge kann in ein Dialogfeld eingegeben werden.
Wenn ein ‹nvar› eingegeben wird, wird das numerische Ergebnis in ‹nvar› platziert.
Wenn numerische Eingabe angefordert wird, wird nur das Drücken von Tasten akzeptiert die 0-9, ”+”, ”-”und”.” sind. Wenn ein ‹svar› angegeben wird, wird der String Ergebnis in ‹svar› platziert. Wenn ein ‹Default_sexp›-Ausdruck angegeben wurde, so wird dieser Ausdruck im Anzeigebereich der Dialogbox angezeigt.
Dieser Ausdruck muss mit dem ‹nvar› oder ‹svar› Typ übereinstimmen.
Wenn der Benutzer auf die Zurück-Taste drückt, wenn das Dialogfeld angezeigt wird sieht der Benutzer die Meldungen:

Input dialog cancelled (Eingabedialog abgebrochen)
Execution halted (Ausführung gestoppt)

, es sei denn, es gibt eine ”OnError:” Anweisung.
Wenn es eine ”OnError:” Anweisung gibt, wird die Ausführung bei der Anweisung die nach ”OnError:” folgt, fortgesetzt.

Inkey$ <svar>

Registriert das Ereigniss des Tastendrückens für die Tasten a-z und 0-9, Leertaste und die D-Pad-Tasten. Der Schlüssel für die Taste wird in ‹svar› zurückgegeben.
Die D-Pad-Tasten werden als ”up”, ”down”, ”left”, ”right” und ”go” registriert. Wenn Sie eine beliebige Taste mit Ausnahme der genannten (also z.B. Großbuchstaben)

drücken, wird die Zeichenkette ”key nn” zurückgegeben. Wobei nn wird der Android Tastencode für diesen Schlüssel sein wird.
Wenn keine Taste gedrückt wurde, wird das Zeichen ”@” zurückgegeben.
Schnelle Tastatureingaben werden gepuffert, für den Fall, dass die Tastenanschläge schneller kommen, als das BASIC! Programm mit ihnen umgehen kann.
=== Bei manchen Android Geräten werden keine Großbuchstaben zurückgegeben===

Text.input <savr>{, <sexp>}

Dieser Befehl ähnelt dem ”Input”, außer dass es zur Eingabe oder zum Bearbeiten Sie einer großen Menge an Text verwendet werden kann. Es
wird ein neues Fenster mit Rollbalken und Volltext-Editierfunktionen geöffnet.
Wenn die optionale ‹sexp› vorhanden ist, dann wird dieser Text in das Eingabefenster für die Bearbeitung geladen. Wenn
‹sexp› nicht vorhanden ist, dann ist der text.input Textbereich leer.
Wenn Sie fertig sind mit der Bearbeitung, drücken Sie die Schaltfläche Finish und der bearbeitete Text wird in ‹svar› zurückgegeben werden. Wenn die BACK-Taste gedrückt wird in der Regel der editierte Text ‹svar› verworfen und das Original zurückgegeben.
Wenn eine Tastatur eingeblendet, wird mit der Back- Taste in der Regel zuerst nur die Tastatur ausgeblendet. ‹sexp› text.
Das folgende Beispiel greift mit dem Beispielprogramm Datei, f01_commands.bas, auf den String s$ zu. Es sendet dann s $
zur Bearbeitung in text.input Fenster. Das Ergebnis der Bearbeitung wird in r$ string zurückgegeben. r$ wird dann im Konsolenfenster angezeigt.

```
grabfile s$, ”../source/ Sample_Programs/f01_commands.bas”
text.input r$,s$
print r$
end
```

TGet <result_svar>, <prompt_sexp>

Simuliert ein Terminal. Der aktuelle Inhalt der Output-Konsole wird angezeigt. Die letzte Zeile die angezeigt wird beginnt mit dem Prompt-String auf das der Cursor folgt. Der Benutzer kann in das Eingabefeld eingeben und dann Enter drücken. Die Zeichen, dass Benutzer eingetippt hat, wird in der Ergebnis-String-Variable zurückgegeben. Die Antwort und wird auf der Output-Konsole angezeigt.

Kb.toggle

Blendet die Software-Tastatur für die Eingabe von Text in das laufende Programm ein. Die Tastenschlüssel können gelesen werden, mit Hilfe des Inkey$- Befehl. Der Befehl

bewirkt nichts bei Geräten mit mechanischer Tastatur oder herausklappbarer Tastatur. Der Befehl sollte gefolgt werden von einer Anweisung ”Pause 1000”, um sicherzustellen, dass die Tastatur-Sichtbarkeit sich ändern kann.

Kb.hide

Blendet die Soft-Tastatur aus.
Hinweis: Wenn die Bildschirmtastatur im Editor angezeigt wird und mit Run das Programm gestartet wird, wird der kb.toggle-Befehl die Tastatur ausblenden. Der folgende Code stellt sicher, dass der Bildschirmtastatur angezeigt wird, unabhängig davon, ob die Tastatur im Editor angezeigt wurde oder nicht.

```
PAUSE 100
KB.HIDE
PAUSE 1000
KB.TOGGLE
```

Arbeiten mit Dateien

BASIC! kann mit Dateien überall auf der SD-Karte arbeiten. BASIC! kann jedoch nicht auf den internen Speicher zugreifen.

Pfad (Paths) - Erklärung

Ein Pfad beschreibt, wo sich eine Datei oder ein Verzeichnis, relativ zu einer Datei oder einem Verzeichniss befindet.
Das Top-Level-Verzeichnis auf der SD-Karte ist: ”/sdcard/”
BASIC! Programmdateien sind in: /sdcard/rfo-basic/source/
BASIC! Data- Dateien sind in:/sdcard/rfo-basic/data/
BASIC! SQLITE Datenbanken sind in:/sdcard/rfo-basic/databases/
Alle BASIC! Datei- Ein- und Ausgabe- (I/O) Befehle setzen einen gewissen Standard-Pfad voraus. Der standardmäßige Pfad lautet ”/scard/rfo-basic/data”, außer für SQLite-Operationen. Wenn Sie mit einer Datei, die nicht in diesem Verzeichnis steht, arbeiten wollen, müssen Sie einen Pfad in das entsprechende Verzeichnis angeben.
Die ”../” Pfad-Notation bedeutet die Rückkehr aus dem aktuellen Verzeichnis um eine Ebene. Der Standardpfad ist der /sdcard/rfo-basic/data/ Pfad. Zurück und dann zum Verzeichnis /sdcard/rfo-basic/source/ geht BASIC! mit ../source/ und erlaubt so einen Einblick in dieses ”/source/” - Verzeichnis. Wenn Sie mit einer Datei im Root-Verzeichnis der SD Card arbeiten wollen, wäre der Weg von dem Standard-Pfad : ../../ .
Der Ausdruck ../../ teilt dem BASIC! mit, zwei Ebenen zurück von /sdcard/rfo-basic/data zu /sdcard/.

Alle die oben genannten Wege bringen Sie zu dem Verzeichnis, in dem Sie eine Datei, lesen, schreiben oder erstellen möchten. Für einen Zugriff auf eine spezielle Datei, müssen Sie den Dateinamen zu dem Pfad hinzufügen.
Der Weg um die Datei, names.txt, in ”/sdcard/rfo-basic/data/” zu lesen, würde der Pfad nur ”names.txt” sein
Der Pfad zu der Programmdatei, sines.bas, in ”/sdcard/rfo-basic/source” wäre ../source/sines.bas
Der Pfad für den Zugriff auf die Musikdatei, rain.mp3, in ”/sdcard/music/”wäre ../../music/rain.mp3

File.Delete <Boolean_nvar>, <Path_sexp>

Die Datei oder das Verzeichnis auf ‹Path_sexp› wird gelöscht, falls sie/es existiert. Wenn es die Datei oder das Verzeichnis das zu löschen ist, vorher nicht gab,
wird die ‹Boolean_nvar› Null enthalten. Wenn die Datei oder das Verzeichnis vorhanden waren und gelöscht wurden, wird die ‹Boolean_nvar› mit nicht Null zurückgegeben werden. Der standardmäßige Pfad lautet ”/sdcard/rfo-basic/data/”

File.Dir <Path_sexp>, Array$ []

Dieser Befehl gibt die Namen der Dateien im Pfad spezifiziert durch den Pfad String-Ausdruck zurück. Die Dateinamen werden in das Array$[] geschrieben. Die Verzeichnis-Namen in der Liste werden mit ”(d)” gekennzeichnet. Die Dateien und Verzeichnisse sind alphabetisch sortiert. Verzeichnisse werden am Anfang der Liste angegeben.
Der standardmäßige Pfad lautet /sdcard/rfo-basic/data/ .

File.Exists <Boolean_nvar>, <Path_sexp>

Dieser Befehl meldet, ob das ‹Path_sexp› Verzeichnis oder die Datei existiert. Wenn das Verzeichnis oder die Datei nicht existiert, erhält die ‹Boolean_nvar› Null. Wenn die Datei oder das Verzeichnis existiert, wird die ‹Boolean_nvar› als Nicht Null zurückgegeben.
Der standardmäßige Pfad lautet /sdcard/rfo-basic/data/

File.Mkdir <Path_sexp>

Bevor Sie ein Verzeichnis verwenden können, muss das Verzeichnis existieren. Der Befehl mkdir dient zum Erstellen von Verzeichnissens.
Der standardmäßige Pfad lautet /sdcard/rfo-basic/data/”
Der Pfad mit dem Sie ein neues Verzeichnis, homes, in /sdcard/rfo_basic/data/ erstellen können wäre /homes/ .
Der Pfad, um ein neues Verzeichnis ”Icons” im Root-Verzeichnis der SD-Karte zu erstellen wäre ”.. /.. / Icons”

File.rename <Old_Path_sexp>, <New_Path_sexp>

Die Datei oder das Verzeichnis Old_Path wird in New_Path umbenannt.
Der standardmäßige Pfad lautet /sdcard/rfo-basic/data/
Mit der Umbenennungs-Operation kann nicht nur der Namen einer Datei oder eines Verzeichnisses geändert werden, sondern man kann sie auch die Datei oder das Verzeichnis in ein anderes Verzeichnis verschieben.
Zum Beispiel:

```
Rename ../../testfile.txt, /data/testfile1.txt
```

wird die Datei, testfile.txt aus "/ SDCard /"entfernen, Sie sie in /sdcard/rfo-basic/data/ verlegen und dabei auch in testfile1.txt umbenennen.

File.root <svar>

Gibt den kanonischen (canonical) Pfad aus dem Dateisystem root /sdcard/rfo-basic/data/ in ‹svar› zurück.

File.Size <size_nvar>, <Path_sexp>

Die Größe der Datei, in Bytes, im Pfad ‹Path_sexp› wird in ‹size_nvar› zurückgegeben.
Der standardmäßige Pfad lautet /sdcard/rfo-basic/data/.

Text File I/O

Die Text-Datei - Eingabe und Ausgabe I/O-Befehle sind ausschließlich für Text (. txt) Dateien zu verwenden. Textdateien bestehen aus Zeilen von Zeichen, die am Ende ein CR (Carriage Return - Wagenrücklauf) und / oder NL (New Line) Zeichen haben.
Die Textdatei Input und Output- Befehle lesen und schreiben ganze Zeilen.
Der standardmäßige Pfad lautet /sdcard/rfo-basic/data/.

Text.open {r|w|a}, <File_table_nvar>, <Path_sexp>

Die durch den ‹Path_sexp› angegebene Textdatei wird geöffnet.
Der erste Parameter beschreibt den I/O-Modus für diese Datei.
r = read (lesen)
w = write - vom Anfang der Datei an schreiben. Schreibt über alle vorhandenen Daten in der Datei.
a = append (anhängen) Das Schreiben beginnt nach der letzten Zeile in der Datei.
Eine Dateinummer wird in ‹File_table_nvar› platziert. Dieser Wert ist für die Verwendung in nachfolgenden text.read oder text.write oder text.close Befehlen notwendig.
Wenn eine Datei so geöffnet werden soll, daß sie für das Lesen nicht existiert, dann wird die ‹File_table_nvar› auf -1 gesetzt. Die BASIC! Programmierer können das zum Überprüfen nutzen und so Fehler an den Benutzer melden.

Text.readln <File_table_nvar>, <Line_svar>

Wenn ‹File_table_nvar› = -1 dann wird ein Laufzeitfehler ausgelöst.
Die nächste Zeile aus der angegebenen, zuvor geöffnete Datei wird in ‹Line_svar› gelesen. Nachdem die letzte Zeile in der Datei gelesen worden ist, werden die Zeichen ”EOF” in ‹Line_svar› platziert.

Dieses Beispiel liest eine gesamte Datei und druckt jede Zeile.

```
Text.open r, file_number, ”testfile.txt”
Do
        Text.read file_number, line$
        Print line$
Until line$ = ”EOF”
Text.close file_number
```

Die Datie wird nicht geschlossen, auch wenn das Ende der Datei gelesen wurde. Ein nachfolgendes Lesen von der Datei bis ”EOF” ist möglich.
Wenn Sie eine Datei gelesen haben, sollten Sie diese schließen.

Text.writeln <File_table_nexp>, < parms identisch zur Ausgabe >

Die Parameter, die den Dateizeiger Tabelle folgen werden analysiert und verarbeitet, genauso wie die PRINT-Befehl-Parameter. Dieser Befehl dient im Wesentlichen zur Ausgabe in eine Datei.
Nachdem die letzte Zeile in die Datei geschrieben worden ist, sollte der Text.close Befehl verwendet werden, um die Datei zu schließen.

Text.position.get <File_table_nvar>, <position_nvar>

Gibt die Position der nächsten Zeile an, die in einer Datei gelesen oder geschrieben werden kann. Die Position dieser ersten Zeile in der Datei ist 1. Auf diese Position ergeben sich aufbauend Inkrementierend die Zeilen die gelesen oder geschrieben werden können. Diese Positionsinformationen können für den Aufbau eines zufälligen Datei-Datenzugriffs verwendet werden. Hinweis: Wenn eine Datei für Append geöffnet wurde, wird die zurückgegebene Position relativ zum Ende der Datei sein. Die Position die angegeben wird, wenn in die erste Zeile nach einer Datei anzufügt wird, wenn die Datei für das Anfügen (append) geöffnet wurde, ist 1. Nutzen sie diese um an neuen Positionen, zu der bekannten Position des am Ende der Datei beim Bau Ihrer zufälligen Access-Tabelle anzufügen.

Text.position.set <File_table_nvar>, <position_nexp>

Legt die Position der nächsten Zeile zu lesen fest. Eine Position, mit dem Wert 1 liest die erste Zeile in der Datei. Text.position.set kann nur für Dateien, die zum Lesen geöffnet wurden angewendet werden. Wenn die Position größer ist als die Anzahl der Zeilen in der Datei, wird die Datei am Ende positioniert werden. Das nächste Lesen würde EOF zurückgeben. Die Position für text.position.get an der EOF zurückgegeben wird, ist die Anzahl der Zeilen in der Datei plus 1.

Text.close <File_table_nvar>

Die zuvor geöffnete Datei ‹File_table_nvar› wird geschlossen.
Hinweis: Es ist besonders wichtig, eine Ausgabe-Datei zu schließen, wenn Sie über 8k Byte auf sie geschrieben haben. Wenn Sie diese nicht schließen, wird die Datei nur die ersten 8K Bytes enthalten.

GrabURL <result_svar>, <url_sexp>

Der gesamte Quelltext der Internet-URL ‹url_sexp› wird in den ‹result_svar› String kopiert. Der Split Befehl kann verwendet werden, um die ‹result_svar› in ein Array von Zeilen aufzuteilen.

GrabFile <result_svar>, <path_sexp>

Der gesamte Inhalt der Datei ‹path_sexp› wird in den ‹result_svar› String kopiert. Der Split Befehl kann verwendet werden, um die ‹result_svar› in ein Array von Zeilen aufzuteilen. Der Befehl kann verwendet werden um den Inhalt einer Textdatei für den direkten Einsatz mit Text.Input einzusetzen.

```
GrabFile text$, MyJournal.txt
Text.Input EditedText$, text$
```

Byte File I/O

Byte-Datei I/O kann verwendet werden, um jede Art von Dateien werden (.txt,. jpg,. pdf, mp3, etc.) zu lesen und zu schreiben. Die Daten werden Byte für Byte gelesen und geschrieben.

Byte.open {r|w}, <File_table_nvar>, <Path_sexp>

Die Datei, des entsprechendens Pfades der im String-Ausdruck angegeben wurde, wird geöffnet. Wenn der Pfad mit ”http ” beginnt, dann wird eine Internet-Datei geöffnet.
Der standardmäßige Pfad lautet /sdcard/rfo-basic/data/
Der erste Parameter beschreibt, den I/O-Modus für diese Datei.
r = read (lesen)

w = write, vom Anfang der Datei an schreiben. Schreibt über alle vorhandenen Daten in der Datei.
A = Append (Anhängen). Startet das Schreiben hinter der letzten Zeile der Datei.
Eine Dateinummer wird in ‹File_table_nvar› festgelegt. Dieser Wert ist für die Verwendung in nachfolgenden text.read oder text.write oder text.close Befehle notwendig.
Wenn eine Datei so geöffnet werden soll, daß sie für das Lesen nicht existiert, dann kann die ‹File_table_nvar› auf -1 gesetzt werden. Die BASIC! Programmierer können das zum Überprüfen vor der Dateierstellung nutzen und so Fehler an den Benutzer melden.

Byte.read <File_table_nvar>, <byte_nvar>

Wenn ‹File_table_nvar› = -1 dann wird ein Laufzeitfehler ausgelöst.
Es wird ein einzelnes Byte aus der Datei gelesen und in ‹byte_nvar› geschrieben.
Nachdem das letzte Byte in der Datei gelesen ist, wird der Wert ‹byte_nvar› als -1 zurückgegeben. Weitere Versuche, aus der Datei zu lesen, verursachen einen Laufzeitfehler.
Dieses folgende Beispiel liest eine gesamte Datei und druckt jedes Byte.

```
Byte.open r, file_number, ”testfile.jpg”
Do
        Byte.read file_number, Byte
        Print Byte
Until Byte ‹ 0
Text.close file_number
```

Byte.write.byte <File_table_nvar>, <byte_nexp>|<sexp>

Wenn der zweite Parameter ein numerischer Ausdruck ist, dann werden die niederwertigen 8 Bits des Wertes der geschrieben werden soll, in die Datei als einziges Byte geschrieben.
Wenn der zweite Parameter ein String-Ausdruck ist, dann wird der gesamte String in die Datei als 8 Bit Bytes geschrieben werden.

Byte.read.buffer <File_table_nvar>, <count_nexp>, <buffer_svar>

Liest die angegebene Anzahl von Bytes aus der Datei, in die Puffer String-Variable.
Wenn das Ende der Datei erreicht ist, wird die Zeichenfolge, len (buf $), wird Null sein.

Byte.write.buffer <File_table_nvar>, <sexp>

Der gesamte Inhalt des String-Ausdrucks wird in die Datei geschrieben.

Byte.position.get <File_table_nvar>, <position_nvar>
Holt sich die Position des nächsten Byte um zu lesen oder zu schreiben. Die Position des ersten Bytes ist gleich 1. Der Positionswert wird um 1 für jedes gelesene oder geschriebene Byte erhöht werden. Die Information über die Position kann für den Aufbau eines zufälligen Datei Datenzugriffs verwendet werden.
Wenn die Datei zum Anhängen geöffnet wird, wird die Position zurückgegeben werden, welche die Länge der Datei sowieso ist.

Byte.position.set <File_table_nvar>, <position_nexp>
Stellt die Position ein, von der aus das nächste Mal aus der Datei gelesen werden soll. Wenn die Position größer ist, als die Position des letzten Bytes der Datei, wird die Position auf das Ende der Datei zeigen.
Dieser Befehl kann nur auf Dateien verwendet werden, die für das Lesen von Bytes geöffnet wurden.

Byte.copy <File_table_nvar>,<output_file_svar>
Der Befehl kopiert die zuvor geöffneten Eingabedatei ‹File_table_nvar› zu der Datei, deren Pfad ‹output_file_svar› ist. Der standardmäßige Pfad lautet /sdcard/rfo-basic/data/.
Wenn ‹File_table_nvar› = -1 dann wird ein Laufzeitfehler ausgelöst.
Die Eingabedatei wird vollständig in die Ausgabedatei kopiert. Beide Dateien werden anschliesend geschlossen.
Sie sollten Byte.Copy benutzen, wenn es um die Eingabe und Ausgabe von Byte (I/O) für den alleinigen Zweck des Kopierens geht. Es geht buchstäblich Tausend mal schneller als mit Byte.read und Byte.write.

Byte.close <File_table_nvar>
Schließt die zuvor geöffnete Datei.

HTML

Einführung
Die BASIC! HTML-Paket wurde entwickelt, um den BASIC! Programmierer zu erlauben Benutzeroberflächen erstellen.
HTML und JavaScript. Die Schnittstelle sorgt für die Interaktion zwischen der HTML-Engine und BASIC! Der HTML-Programmierer kann JavaScript verwenden, um Nachrichten an den BASIC senden!. Die HTML-Engine unterstützt auch

Anwender Ereignisse wie die BACK-Taste, Hyperlink Transfers, Downloads, Formulardaten und Fehlermedungen.
Das Demo-Programm, f37_html_demo.bas, mit den Demo-HTML-Dateien htmlDemo1.html und htmlDemo2.html kombiniert, veranschaulicht die verschiedenen Befehle und Möglichkeiten. Der Inhalt von allen drei Dateien ist in Anhang B dieses Dokuments aufgeführt. Sie werden auch mit dem BASIC! apk ausgeliefert. Es wird dringend empfohlen, dass alle drei Dateien sorgfältig studiert werden, um diese Schnittstelle besser zu verstehen,.
Ein weiteres Demo-Programm f38_html_edit.bas dient zum Bearbeiten von HTML-Dateien. Um das Programm zu verwenden, führen Sie es aus und geben Sie den HTML-Dateinamen ohne die Endung html ein. Das Programm fügt das ". html" an.

Befehle

Html.open {} <Show_status_bar_nexp>

Dieser Befehl muss vor der Verwendung des HTML-Schnittstelle ausgeführt werden. Mit dem optionalen numerischen Ausdruck wird bei Zugriff die Statusleiste angezeigt, wenn der Wert nicht 0 ist. Der Standardwert ist ohne Statusleiste.

Html.load.url <file_sexp>

Lädt und zeigt die Datei die im String angegeben wurde. Die Datei kann im Internet oder auf Ihrem Android-Gerät sein. In jedem Fall muss die gesamte URL angegeben werden.
Der Befehl:

```
html.load.url "http://laughton.com/basic/"
```

Lädt und zeigt die BASIC! Homepage.
Der Befehl:

```
html.load.url "file:///sdcard/rfo-basic/data/htmlDemo1.html"
```

Lädt und zeigt die HTML-Datei der angegebenen Position.
Wenn Sie die Taste Zurück auf der geladenen Seite drücken, wird der HTML-Viewer geschlossen und die BASIC! Output-Konsole wird angezeigt. Wenn mit der ursprünglich geladenen Seite Links zu anderen Seiten gegangen werden und dann die BACK-Taste gedrückt wird, muss der BASIC! Programmierer entscheiden, was zu tun ist.

html.load.string <html_sexp>

Lädt und zeigt die HTML-Seite die im String-Ausdruck enthalten ist. Die Basis für diese HTML- Seite ist:

/sdcard/rfo-basic/data/

html.post url$, list

Führen Sie einen Post-Befehl zu einem Internet-Standort oder einer HTML-Datei aus.

url$ ist ein String-Ausdruck der die URL angibt, die den Beitrag akzeptiert.
List ist ein Zeiger auf einen String-List, die Name / Wert-Paare enthält, welche der Post Befehl benötigt.

Html.get.datalink <data_svar>

Ruft die nächste Datalink- Zeichenfolge vom Datalink-Puffer ab. Wenn keine Datalinked Daten vorhanden sind, dann wird zurückgegebenen Datenstring eine leere Zeichenfolge ("") sein. Sie sollten eine Schleife programmieren, die auf Daten wartet:

```
do
html.get.datalink data$
until date$ <> ""
```

Die zurückgegebenen Daten-Zeichenkette wird immer mit einer bestimmten Gruppe von vier Zeichen beginnen. Diese vier Zeichen identifizieren, die Rückkehr des DATALINK-Datentyp. Den meisten Typ-Codes folgen in irgendeiner Art Daten.
Die Codes sind:

BAK: Die BACK-Taste wurde gedrückt. Die Daten sind entweder "1" oder "0"
Wenn die Daten "0" sind drückte der Benutzer Back im Startbildschirm. Ein weiteres zurückgehen ist nicht möglich, bevor html geschlossen wurde.
Wenn die Daten "1" sind, dann ist ein zurückgehen möglich. Der BASIC! Programmierer sollte den Befehl html.go.back nutzen, wenn zurück gewünscht wird.

LNK: Der Benutzer hat einen Hyperlink angetippt. Die verlinkte URL wird zurückgegeben. Die Übertragung auf die neue URL hat nicht stattgefunden. Der BASIC! Programmierer muss "html.load.url" mit der zurückgegebenen URL (oder einer anderen ausführbaren URL) ausführen, damit eine Übertragung erfolgen kann.

ERR: Irgendeine Art von schwerwiegender Fehler ist aufgetreten. Die Fehlerbedingung wird zurückgegeben. Dieser Fehlercode schließt immer die HTML-Engine. Die BASIC! Output-Konsole wird angezeigt.

FOR: Der Benutzer hat die Schaltfläche Senden auf einem Formular mit action = "FORM" gedrückt. Das Formular Name / Wert-Paar wird zurückgegeben. Die HTML-Engine wird automatisch zu dieser Zeit geschlossen. Die BASIC! Programmierer können auf Wunsch eine neue HTML-Session öffnen und eine HTML-Antwort-Seite anzeigen.

DNL: Der Benutzer hat einen Link, der einen Download anfordert angeklickt. Die Download-URL wird mitgeliefert. Es ist Aufgabe des BASIC! Programmierer, sich um den Download zu kümmern.

DAT: Der Benutzer hat eine Aktion durchgeführt und hat damit JavaScript-Code verursacht, um Daten zu senden an BASIC! mittels der Sicherungsschicht. Die

JavaScript-Funktion für das Senden der Daten ist:

```
<script type="text/javascript">
funktion doDataLink (data) {
Android.dataLink (data);
}
</ Script>
```

Html.go.back
Gehen Sie einen HTML-Bildschirm zurück, wenn möglich.

Html.go.forward
Gehe einen HTML-Bildschirm vorwärts , wenn möglich.

Html.close
Schließt die HTML-Engine und die Anzeige.

Html.clear.cache
Löscht den HTML-Cache.

HTML.clear.history
Löscht die HTML-Geschichte.

TCP/IP Sockets (Buchsen)
TCP/IP-Sockets stehen für die Übertragung von Informationen von einem Punkt im Internet zu einem anderen zur Verfügung. Dort gibt es zwei Arten von TCP / IP-Sockets: Server und Clients. Die Clients müssen mit den Servern kommunizieren. Server müssen mit den Clients reden. Clients können nicht mit Clients und Server können nicht mit Servern kommunizieren.
Jeder Client-und Server-Paar haben ein Protokoll vereinbart. Dieses Protokoll legt fest, wer zuerst spricht, sowie die Bedeutung und die Reihenfolge der Nachrichten, die zwischen ihnen fließen. Die meisten Menschen, die eine FTP/IP-Socket verwenden, nutzen eine Client-Socket zur Ausgabe von Nachrichten mit einem vorhandenen Ausgabe- Server mit einem vorgegebenen Protokoll. Ein einfaches Beispiel hierfür ist das Beispielprogramm f31_socket_time.bas.
Dieses Programm verwendet eine TCP / IP-Client-Socket, um die aktuelle Zeit von einem der vielen Zeit-Server in den USA zu erhalten.
Ein TCP/IP Server lässt sich mit BASIC! einstellen; aber es ist etwas kompliziert. Die Fähigkeiten der individuellen drahtlosen Netzwerke variieren. Einige drahtlose Netzwerke ermöglichen Server. Die meisten tun dies nicht. Server können in der Regel über WiFi oder Ethernet Local Area Networks (LAN) betrieben werden.

Wenn Sie einen Server einrichten wollen, ist es der naheliegendste Weg den Server innerhalb eines LAN zu etablieren. Sie müssen ein Port- Tunneling (Weiterleitung) von der LAN's externen Internal-IP zu der LAN-Geräte IP anbieten.
Sie müssen in der Lage sein, das Programm (Setup) des LAN-Router aufzurufen, um dies zu tun.
Clients, egal ob innerhalb eines Server- LAN's oder vom Internet, sollten zum LAN und externer IP-Adresse durch Nutzung des vorher festgelegten, getunnelten Port verbunden werden. Diese externe WAN-IP kann durch Verwendung von

Graburl ip$, "http://automation.whatismyip.com/n09230945.asp"

gefunden werden.
Dies ist nicht der gleiche IP, die durch das Ausführen von socket.myip auf der Server-Einheit erhalten werden würde.
Hinweis: Die angegebenen IPs müssen keine numerische Form haben. Sie können in der Namen- Form vorliegen.
Das Beispiel-Programm f32_tcp_ip_sockets.bas, demonstriert die socket- Befehle für einen Server der in Verbindung mit einem Client arbeitet. Sie benötigen zwei Android-Geräte, um dieses Programm auszuführen.

TCP/IP-Client-Socket-Befehle

Socket.client.connect <server_ip_sexp>, <port_nexp>

Erstellt eine Client TCP/IP-Socket und versucht, den Server, dessen IP wird durch die Server-IP angegeben wurde, zu erreichen. Der String-Ausdruck benutzt den Port, der durch den numerischen Port- Ausdruck angegeben ist. Dieser Befehl wird nichts zurückgeben, bis die Verbindung hergestellt wurde oder ein Fehler erkannt wird. Wenn das Gerät an der angegebenen IP nicht reagiert, wird der Befehl Timeout nach ein paar Minuten ausgegeben.

Socket.client.read.ready <nvar>

Wenn das zuvor erstellte Client-Socket keine Zeile zum Lesen von socket.client.read.line erhält, dann wird ‹nvar› wird als Null zurückgegeben. Andernfalls wird ein Nicht-Null-Wert zurückgegeben.
Der socket.client.read.line Befehl gibt nichts zurück, bis eine Zeile vom Server empfangen wurde. Dieser Befehl kann verwendet werden, um in Ihren Programm ein time out zu erreichen, wenn eine Zeile nicht innerhalb einer vorgegebenen Zeitspanne eingegangen ist. So können sie sicher sein, dass socket.client.read.line eine Zeile mit einer Reihe von Daten zurückgegeben hat, wenn dieser Befehl einen Wert ungleich Null ergibt.

Socket.client.read.line <line_svar>

Liest eine Zeile aus dem zuvor kontaktierten Server und setzt die Zeile in die String-Variable ‹line_svar›. Der Befehl gibt nichts zurück, bis der Server eine Zeile gesendet hat. Um eine unendliche Verzögerung durch das Warten auf den Server für das Senden einer Zeile, zu vermeiden, kann der socket.client.read.ready Befehl wiederholt mit Timeouts ausgeführt werden.

Socket.client.read.file <fw_nexp>

Eine Datei die von einem Server übertragen wurde, wird gelesen und in die Datei geschrieben, die vom zuvor ausgeführten byte.open Schreib-Befehl abgeleitet wurde.
Zum Beispiel:

```
Byte.open w, fw, ”image.jpg”
Socket.client.read.file fw
Byte.close fw
```

Socket.client.write.line <line_sexp>

Sendet den String-Ausdruck, zu dem zuvor kontaktierten Server als UTF-16 Zeichen. Zeilenende- Zeichen werden am Ende der Zeile durch die TCP/IP-Socket-Server-Befehle hinzugefügt.

Socket.client.write.bytes <sexp>

Sendet den String-Ausdruck, zu dem zuvor kontaktierten Server als UTF-8 Zeichen. Es werden keine Zeilenendezeichen der Zeile von BASIC! hinzugefügt. Wenn Sie eine CR oder LF-Zeichen benötigen, müssen Sie ‹sexp› um diesen Teil ergänzen. Beachten Sie, dass wenn socket.server.read.line verwendet wird, um diese Bytes zu empfangen, der read.line Befehl nichts zurückgibt, bis er ein LF- Zeichen Line Feed (10, 0x0A) erhält.

Socket.client.write.file <fr_nexp>

Eine vorher zum Lesen geöffnete Datei von byte.open wird an den Client übertragen werden.
Zum Beispiel:

```
Byte.open w, FR, ”bild.jpg”
Socket.client.write.file fr
Byte.close fr
```

TCP / IP-Socket-Server-Befehle

Socket.myip <svar>
Gibt die IP des Gerätes in ‹svar› zurück.
Wenn sich das Gerät an einem WiFi-oder Ethernet-LAN befindet, dann wird diese IP, die LAN-Gerät- IP sein.
Hinweis: Dieser externe oder WAN-IP finden Sie auch mit:
Graburl ip$, ”http://automation.whatismyip.com/n09230945.asp”

Socket.server.create <port_nexp>
Richtet einen Server ein, der durch den mit dem numerischen Ausdruck spezifizierten Port, hören wird.

Socket.server.connect
Weist den zuvor erstellten Server an, eine Verbindung mit dem nächsten Client in der Warteschlange zu akzeptieren.
Dieser Befehl wird nichts zurückgeben, bis eine Verbindung mit einem Client hergestellt wird.

Socket.server.read.ready <nvar>
Wenn die vorher akzeptierte Client-Socket keine Zeile zum Lesen mit socket.server.read.line geschickt hat, dann wird ‹nvar› als Null zurückgegeben.
Andernfalls wird ein Nicht-Null-Wert zurückgegeben.
Der socket.server.read.line Befehl gibt nichts zurück, bis eine Zeile vom Client empfangen wurde. Dieser Befehl kann verwendet werden, um in Ihren Programm ein time out zu erreichen, wenn eine Zeile nicht innerhalb einer vorgegebenen Zeitspanne eingegangen ist. So können sie sicher sein, dass socket.client.read.line eine Zeile mit einer Reihe von Daten zurückgegeben hat, wenn dieser Befehl einen Wert ungleich Null ergibt.

Socket.server.read.line <svar>
Liest eine Zeile aus dem zuvor kontaktierten Client und setzt die empfangene Zeile in die ‹svar› String-Variable.
Der Befehl wird nichts zurückgeben, bis der Client eine Linie gesendet hat. Um eine unendliche Verzögerung durch Warten auf das Senden durch
den Client zu vermeiden, kann der socket.server.read.ready Befehl wiederholt mit Timeouts ausgeführt werden.

Socket.server.write.line <sexp>
Sendet den String-Ausdruck, zu dem zuvor kontaktierten Client als UTF-16 Zeichen.

Zeilenendezeichen werden an das Ende der Zeile geschrieben.

Socket.server.write.bytes <sexp>
Sendet den String-Ausdruck zu den zuvor kontaktierten Client als UTF-8 Zeichen. Es werden keine Zeilenendezeichen
von BASIC! hinzugefügt. Wenn Sie ein CR aus LF-Zeichen benötigen, müssen Sie ‹sexp› um diesen Teil ergänzen. Beachten Sie, dass bei der Anwendung des socket.client.read.line Befehls um Bytes zu empfangen, der Befehl read.line nichts zurückgeben wird, bis er ein Line Feed (10, 0x0A) Zeichen erhält.

Socket.server.write.file <fr_nexp>
Eine vorher mit byte.open geöffnete Datei wird gelesen und an den Client übertragen. Beispiel:

```
Byte.open w, fr, ”image.jpg”
Socket.server.write.file fr
Byte.close fr
```

Socket.server.disconnect
Die Verbindung mit dem zuvor verbundenen Client wird geschlossen. Eine neue socket.server.connect Verbindung kann
aufgebaut werden, um mit dem nächsten Client in der Warteschlange zu verbinden.

Socket.server.close
Der zuvor erstellte Server wird geschlossen. Jeder aktuell verbundene Client wird getrennt.

Socket.server.client.ip <nvar>
Gibt die IP des Clients zurück, der aktuell mit dem Server verbunden ist.

FTP-Client
Diese FTP-Befehle implementieren einen FTP-Client

ftp.open <url_sexp>, <port_nexp>, <user_sexp>, <pw_sexp>
Verbindet mit angegebener URL und Port. Logt bei dem Server mit dem angegebenen Benutzernamen und Kennwort ein. Zum Beispiel:

```
ftp.open ”ftp.laughton.com”, 21, ”basic”, ”basic”
```

ftp.close
Trennt die Verbindung zum FTP-Server.

ftp.put <source_sexp>, <destination_sexp>

Läd (Upload) die angegebene Quelldatei hoch, auf die angegebene Ziel-Datei auf den angeschlossenen FTP-Server. Die Quelldatei wird relativ zum Verzeichnis "/sdcard /rfo-basic/data/" angegeben. Wenn Sie eine BASIC!- Quelldatei hochladen wollen, würde der Dateinamen-Pfad: ".../source/xxxx.bas" sein.
Die Zieldatei ist relativ zum aktuellen Verzeichnis auf dem Server anzugeben. Wenn Sie ein Upload in ein Unterverzeichnis des aktuellen Arbeitsverzeichnisses wollen, geben Sie den Pfad zu diesem Verzeichnis an. Zum Beispiel, wenn es ein Unterverzeichnis mit dem Namen "etc" ist, dann ist der Dateiname, "/etc/name" für die Datei die in dieses Unter-Verzeichnis hochgeladen werden soll.

ftp.get <source_sexp>, <destination_sexp>

Die Quelldatei auf dem kontaktierten FTP-Server, wird auf die angegebene Ziel-Datei auf dem Android- Gerät heruntergeladen. Sie können ein Unterverzeichnis in der Server-Quelldatei Zeichenkette angeben. Der Zieldatei Pfad ist relativ zu"/sdcard/rfo-basic/data/" anzugeben. Wenn Sie ein BASIC! Quelldatei herunterladen wollen, würde der Pfad ".. /source/xxx.bas" sein.

ftp.dir <list_navar>

Erstellt eine Liste der Dateien des aktuellen Arbeitsverzeichnises und platziert diese Liste in eine BASIC! List data Struktur. Verzeichnisse werden durch die angehängten Zeichen "(d)" nach dem Namen gekennzeichnet. Das folgende Codebeispiel kann verwendet werden, um die Dateinamen in dieser Liste anzuzeigen:

```
ftp.dir file_list
list.size file_list,size
for i = 1 to size
      list.get file_list,i,name$
      print name$
next i
```

ftp.cd <new_directory_sexp>

Wechselt vom aktuellen Verzeichnis an das angegebene neue Verzeichnis.

ftp.rename <old_filename_sexp>, <new_filename_sexp>

Benennt den angegebenen alten Dateinamen in den angegebenen neuen Dateinamen um.

ftp.delete <filename_sexp>

Löscht die angegebene Datei.

ftp.rmdir <directory_sexp>
Entfernt (löscht) das angegebene Verzeichnis, aber nur wenn das Verzeichnis leer ist.

ftp.mkdir <directory_sexp>
Erstellt ein neues Verzeichnis mit dem angegebenen Namen.

Bluetooth
BASIC! beinhaltet Bluetooth Befehle in einer Weise, welche die Übertragung von Datenbytes zwischen einem Android Gerät und einem anderen Gerät (das sowohl ein weiteres Android Gerät sein kann oder auch nicht) ermöglicht.
Bevor BASIC! Bluetooth-Befehle ausgeführt werden können, sollten Sie in der Android Bluetooth Software über die ”Setting” Einstellungen Bluetooth aktivieren und ”pair” (Kopplung) mit jedem Gerät ermöglichen, mit dem Sie planen zu kommunizieren.
Wenn Bluetooth geöffnet wird (bt.open), geht es in den Listen-(Hören) Modus. In diesem Modus wird es auf ein Gerät warten um sich damit zu verbinden. Sie können diesen Modus, in den Connect-Modus ändern, durch die Ausführung des bt.connect Befehls. Nach der Ausführung des bt.connect Befehls erhalten Sie eine Liste der gekoppelten Bluetooth-Geräte und werden gefragt welches Sie auswählen wollen, um eine Verbindung herzustellen. BASIC! wird versuchen, an dieses Gerät anzukoppeln, sobald es aktiviert ist.
Sie sollten den Zustand des Bluetooth mit dem Befehl br.state überwachen. Dieser Befehl wird über den Status von Zuhören, Ankoppeln (Connecting) und Verbundensein (Connected) berichten. Sobald Sie einen ”Connected”-Bericht, erhalten können Sie mit Bytes lesen und Bytes schreiben an das angeschlossene Gerät beginnen.
Sie können Bytes in ein angeschlossenes Gerät mit dem Befehl bt.write schreiben.
Die Daten werden vom angeschlossenen Gerät mit dem bt.read.bytes Befehl zu lesen sein, jedoch vor dem Ausführen von bt.read.bytes, müssen Sie herausfinden, ob es zu lesenden Daten gibt. Hierzu verwenden Sie den bt.read.ready Befehl.
Sobald die Verbindung steht, sollten Sie auch weiterhin den Status überwachen (mit Hilfe bt.status), um sicherzustellen, dass das angeschlossene Gerät verbunden bleibt.
Wenn Sie mit einer bestimmten connection oder mit Bluetooth generell fertig sind, führen sie bt.close aus.
Das Beispiel-Programm, f35_bluetooth, ist ein funktionierendes Beispiel für Bluetooth mit zwei Android-Geräten in einer Chat-Typ Application.

Bt.open
Öffnet Bluetooth in Listen (Hören) Modus. Wenn Sie noch nicht Bluetooth-fähig sind (mit

den Einstellungen für die Android
Application) dann werden Sie danach gefragt werden, ob sie Bluetooth aktivieren möchten.

Bt.close

Schließt das zuvor geöffnete Bluetooth- oder die Verbindung. Bluetooth wird automatisch geschlossen werden, wenn die
Programmausführung endet.
BASIC!-Befehle um zu einem bestimmten Gerät zu verbinden. Die Ausführung dieses Befehls führt zu einer Liste der gekoppelten
Geräte, die angezeigt werden. Wenn eines dieser Geräte ausgewählt wurde bekommt es im bt.status ”Connecting”
bis das Gerät verbunden (Connected) ist.

Bt.reconnect

Dieser Befehl wird versuchen, an ein Gerät noch einmal anzukoppeln, mit dem es zuvor verbunden war. Das Gerät muss zuvor (während dieses Programmablaufs) mit dem ”bt.connect: Befehl angeschlossen gewesen sein. Der Befehl kann nicht verwendet werden, um mit einem Gerät, das nur in den Listen-(Hören) Modus geschaltet wurden verbunden zu werden. Überwachen Sie den Bluetooth-Status für connected (3) nach dem Ausführen von bt.reconnect.

Bt.status <nvar>

Ruft die aktuelle Bluetooth-Verbindungsstatus auf und legt den Wert in die numerische Variable.

0 = Nothing going on
1= Listening
2= Connecting
3 = Connected

Bt.write <parms wie Ausgabe>

Schreibt eine Textzeile in die Bluetooth-Verbindung. Die Parameter sind identisch mit den Ausgabeparameter. Dieser Befehl ist im Wesentlichen für die Ausgabe in Bluetooth-Verbindungen.

Bt.read.ready <navar>

Gibt in der numerischen Variablen die Anzahl der Nachrichten die zum Lesen bereit sind, an. Wenn der Wert größer ist als

Null, sollte die Nachricht gelesen werden können, bis die Warteschlange leer ist.

OnBTReadReady:

Wenn eine Bluetooth Nachricht fertig empfangen wurde (BT.READ.READY wäre ein Rückgabewert ungleich null) wird das aktuell laufende Programm unterbrochen und die Ausführung wird bei der Anweisung nach diesem Ausdruck fortgesetzt. Sie können dann die Nachricht lesen und weiterverarbeiten. Wenn Sie fertig sind, können Sie mit dem Bt.onReadReady.Resume-Befehl das unterbrochene Programm wieder ausführen.

Bt.onReadReady.Resume

Setzt die Ausführung des Programms an der Stelle fort, wo es durch das Ereignis Bluetooth Read Ready unterbrochen wurde.

Bt.read.bytes <svar>

Der nächste verfügbare Nachricht wird in die angegebene String-Variable platziert. Wenn es keine Nachricht gibt, dann wird die String-Variable wird mit einer leeren Zeichenfolge ("") zurückgegeben.

Bt.device.name <svar>

Gibt den Namen des angeschlossenen Gerätes in der String-Variable. Ein Laufzeitfehler wird generiert, wenn kein Gerät (Status = 3) connected ist.

Bt.set.uuid <sexp>

Ein Universally Unique Identifier (UUID) ist eine standardisiertes 128-Bit-Format für eine String-ID zur eindeutigen Identitäts- Information. Das Besondere einer UUID ist, dass sie groß genug ist, um eine zufällige Auswahl zu ermöglichen ohne gleich mit einer anderen zu kollidieren. In diesem Fall nützt es zur eindeutigen Identifizierung Ihrer Anwendung als Bluetooth-Dienst. Um eine eindeutige UUID zu erhalten für Ihre Anwendung, können Sie diese mit einem der vielen UUID-Zufalls-Generatoren im Internet erstellen.
Viele Geräte haben gemeinsame UUIDs für die jeweilige Anwendung. Die Standard-BASIC! UUID ist die Standard Serial Port Profile (SPP) UUID: "00001101-0000-1000-8000-00805F9B34FB"
Sie können die Standard-UUID mit diesem Befehl ändern.
Einige Informationen zu 16 Bit und 128 Bit UUIDs sind zu finden unter:
http://farwestab.wordpress.com/2011/02/05/some-tips-on-android-and-bluetooth/

Verschiedene Befehle

Browse <url_sexp>

Wenn ‹url_sexp› mit "http... " dann wird die Internet-Seite die von ‹url_sexp› angegeben

wird, geöffnet und angezeigt. Wenn ‹url_sexp› mit ”file:///SDCard/” beginnt dann wird die Datei von der Anwendung, ThinkFree, Office geöffnet werden. Die Dateitypen, die ThinkFree Office öffnen kann sind .txt, .pdf, .doc,.xls, .rtf.
Wenn Ihr Android-Gerät nicht bereits über ThinkFree Office verfügt, können Sie ThinkFree Office im Android Market finden. Es ist kostenlos.
Hinweis: Sie können auch HTML-Befehle nutzen um Webseiten sich auf dem Gerät oder im Web befinden, anzuzeigen (und damit zu interagieren).

Swap <nvar_a> | <svar_a>, <nvar_b>, <svar_b>
Die Werte und in ”a” und ”b” numerische oder String-Variablen werden getauscht.

Clipboard - Zwischenablage

Clipboard.get <svar>
Kopiert den aktuellen Inhalt der Zwischenablage in ‹svar›

Clipboard.put <sexp>
Platziert ‹sexp› in die Zwischenablage.

Echo.on
Das Gleiche wie debug.echo.on. Siehe debug.echo.on für Details.

Echo.off
Das Gleiche wie debug.echo.off. Siehe debug.echo.off für Details.

Encryption - Verschlüsselung
Verschlüsselt und entschlüsselt einen String mit Hilfe eines mitgelieferten Passwortes.

Encrypt <pw_sexp>, <source_sexp>, <encrypted_svar>
Die ‹source_sexp›Zeichenkette wird mittels ‹pw_sexp› Passwort verschlüsselt. Das verschlüsselte Ergebnis wird in ‹encrypted_svar› geschrieben.

Decrypt <pw_sexp>, <encrypted_svar>, <decrypted_svar>
Die verschlüsselten Daten in ‹encrypted_svar› werden mit dem Passwort‹pw_sexp› entschlüsselt . Die entschlüsselten Ergebnisse werden in ‹decrypted_svar› abgelegt.

Text To Speech
Die Sprache entspricht der aktuellen Standard-Sprache des Gerätes. Der Standardwert Sprache wird in Android gesetzt mit:

1. Starten Sie die Anwendung Einstellungen
2. Wählen Sie Voice-Input & Output
3. Wählen Sie Text-to-Speech-Einstellungen

4. Wählen Sie die Sprache

Tts.init
Dieser Befehl muss ausgeführt werden, bevor Sie sprechen können.

Tts.speak <sexp>
Spricht den String-Ausdruck. Der Befehl wird nichts zurückgeben, bis die Zeichenkette vollständig gesprochen worden ist.

Sprache in Text (Spracherkennung)
Die Spracherkennungs- Funktion auf Android verwendet die Google-Server um die Erkennung durchzuführen. Dies bedeutet, dass Sie mit dem Internet verbunden und in Ihrem Google-Konto angemeldet sein müssen, damit dieses Feature funktioniert.
Es gibt zwei Befehle zur Sprache in Text: STT.LISTEN (Hören) und STT.RESULTS.
Das STT.LISTEN Kommando startet den Erkennungsvorgang der Stimme mit einem Dialogfeld. Die STT.RESULTS berichtet die Interpretation der Stimme mit einer Liste von Zeichenfolgen. Das Spracherkennungs- Verfahren ist für den Grafikmodus, HTML und einfache Konsolenausgabe Modus unterschiedlich.

Befehle

STT.LISTEN
Startet das Stimme erkennen und verarbeiten durch ein Dialogfeld ”Jetzt sprechen”.
Beginnen Sie zu sprechen.
Die Erkennung wird beendet, wenn eine Sprechpause gemacht wird.
STT.RESULTS sollte als Nächstes ausgeführt werden.
Hinweis: STT.LISTEN wird nicht m HTML-Modus ausgeführt.

STT.RESULTS <string_list_ptr_nvar>
Der Befehl muss nicht erst nachdem ein STT.LISTEN ausgeführt wurde, ausgeführt werden (es sei denn, im HTML-Modus).
Die Erkennung gibt mehrere Varianten an von dem, was es zu hören schätzt, es als eine Liste von Zeichenfolgen. Die erste Zeichenfolge in der Liste ist die beste Schätzung.

Konsolen-Modus
Der folgende Code veranschaulicht den Befehl in der Output-Konsole (kein HTML-Modus und kein Grafik-Modus):

```
PRINT “Starting Recognizer”
STT.LISTEN
STT.RESULTS  theList
LIST.SIZE theList, theSize
```

```
FOR k =1 to theSize
        LIST.GET theList, k, theText$
        PRINT theText$
NEXT k
END
```

Grafik-Modus

Diese Befehlssequenz sollte im Grafikmodus verwendet werden. Der Grafik-Modus existiert nach GR.OPEN und vor dem GR.CLOSE. (Hinweis: Der Grafik-Modus endet vorübergehend nach GR.FRONT 0. Verwenden Sie den Konsolenmodus, wenn Sie, GR .FRONT 0 aufgerufen haben.

Der Hauptunterschied ist, dass GR.RENDER nach STT.LISTEN und vor STT.RESULTS aufgerufen werden muss.

```
PRINT “Starting Recognizer”
STT.LISTEN
GR.RENDER
STT.RESULTS  theList
LIST.SIZE theList, theSize
FOR k =1 to theSize
        LIST.GET theList, k, theText$
        PRINT theText$
NEXT k
END
```

HTML-Modus

Diese Befehlsfolge wird im HTML-Modus verwendet. Der HTML-Modus existiert nach HTML.OPEN und vor HTML.CLOSE.

Der Hauptunterschied ist, dass die STT.LISTEN-Befehl wird nicht im HTML-Modus verwendet wird. Die STT.LISTEN-Funktion erfolgt durch eine Rücksendung der Zeichenfolge ”STT” als HTML-Datalink. Das Senden von ”STT” durch den Datalink führt dazu dass ”Jetzt sprechen - Speak Now” im Dialogfeld angezeigt wird.

Wenn die Datalink ”STT” Zeichenfolge vom BASIC! Programm empfangen wird, kann der STT-RESULT Befehl wie oben dargestellt, ausgeführt werden.

Die Beispieldatei, f37_html_demo.bas, veranschaulicht die Verwendung der Spracherkennung im HTML-Modus.

Zeitgeber-Interrupts und Befehle

Einen Timer (Zeitgeber) kann festgelegt werden, um die Ausführung des Programms in bestimmten Abständen nach eingestellter Zeit zu unterbrechen. Wenn die Unterbrechung (Interrupt) auftritt, wird die Programmausführung, die Anweisungen nach dem OnTimer:label übertragen. Wenn Sie getan haben, was Sie tun mussten, um dieses Timer-Ereignis zu behandeln, verwenden Sie den Timer.Resume-Befehl, um die

Ausführung des Programms an der Stelle fortzusetzen, wo der Timer-Interrupt aufgetreten ist.

Timer.Set <interval_nexp>

Legt einen Zeitgeber fest, der immer wieder die Ausführung des Programms nach der angegebenen Zeitspanne unterbricht. Die Intervall-Zeiteinheiten sind Millisekunden. Das Programm muss ein Label ”onTimer” enthalten, wenn dieser Befehl ausgeführt wird.

onTimer:

Das Label nach dem der Timer Interrupt Code platziert werden kann.

Timer.Resume

Bewirkt, dass die Ausführung des Programms an dem Punkt wieder aufzunehmen, wo der Timer-Interrupt aufgetreten ist.

Timer.Clear

Löscht den wiederholten Zeitgeber. Keine weiteren werden der Zeitgeber-Interrupts auftreten.

Beispiel-Code

```
n=0
TIMER.SET 2000

DO
UNTIL n=4
TIMER.CLEAR
PRINT “Timer cleared. No further interrupts.”
DO
UNTIL O

ONTIMER:
n = n + 1
PRINT n*2; “ seconds”
TIMER.RESUME
```

Device <svar>

Liefert Informationen über das Android-Gerät in die String-Variable. Die Informationen sind: das Geräte- Model, Device Type und OS.

Include FileNamePath

Fügt eine weitere BASIC! Programm-Datei an dieser Stelle im Programm ein. Der FileNamePath wird ohne Anführungszeichen angegeben. Der
Befehl wird ausgewertet, bevor das Programm startet und der Pfad kann deshalb kein String-Ausdruck werden.
”Include/functions/DrawGraph.bas” wird den Code aus Datei ”DrawGraph.bas” aus dem Verzeichnis, ”/sdcard/rfo-basic/source/functions/” in das Programm einfügen.

Pause <ticks_nexp>

Stoppt die Abarbeitung des BASIC! Programm für ‹ticks_nexp› Millisekunden. Einer Millisekunde = 1/1000 einer Sekunde. Pause 1000 wird die Programmausführung für eine Sekunde anhalten.

Popup <message_sexp>, <x_nexp>, <Y_nexp>, <duration_nexp>

Öffnet ein kleines Nachrichten Popup-Fenster für eine begrenzte Dauer.
Die Nachricht ist ‹message_sexp›.
Die Dauer der Nachricht ist entweder 2 Sekunden oder 4 Sekunden. Wenn ‹duration_nexp› = 0 ist die Dauer 2 Sekunden. Wenn ‹duration_nexp› ‹› 0 wird die Dauer 4 Sekunden sein. Der Standard-Anzeigeort für das Popup-Fenster ist das Zentrum des Bildschirms. Die ‹x_nexp› und ‹Y_nexp› Werte geben die Möglichkeit zur Verschiebung aus dem Zentrum. Die Werte können auch negativ sein.

Select <selection_nvar>, < Array$[]>|<list_nexp>, <message_sexp> {,<press_nvar>}

Das Select-Kommando erzeugt einen neuen Bildschirm mit einer Liste von Auswahlmöglichkeiten für den Benutzer. Wenn der Benutzer eine Bildschirmzeile antippt, wird der Array-Index für diese Zeile zurückgegeben werden.
‹selection_nvar› ist die numerische Variable, in welche der Auswahl Array-Index zurückgegeben wird.
‹Array$[]› ist ein String-Array, das die Liste der Elemente enthält die ausgewählt wurden. Das Array ist ohne Index spezifiziert, muss aber zuvor dimensioniert oder mit Array.load geladen werden.
Als Alternative zu einem Array kann ein String des Typs Liste in ‹list_nexp› angegeben werden.
‹message_sexp› ist ein String-Ausdruck, der in der Titelleiste am oberen Rand der Markierung am Bildschirm platziert wird.
Er kann auch in einer kurzen Popup-Meldung verwendet werden, es sei denn die Meldung ist eine leere Zeichenkette (""). In diesem Fall ergibt sich kein Popup Fenster.
Die ‹press_nvar› ist optional. Falls vorhanden, kann die Art des Drückens durch den Benutzer, lang oder kurz, in ‹press_nvar› zurückgegeben werden. Der zurückgegebene Wert ist 0, wenn der Benutzer das Element mit einem kurzen Antippen ausgewählt hat. Der Wert der zurückgegeben wird 1, wenn der Benutzer das Element mit einem langen Druck ausgewählt hat.

Split <result_Array$[]>, <source_sexp>, <test_sexp>

Die Zeichenkette ‹spource_sexp› wird in mehrere Strings, die in ‹result_Array$[]›

platziert sind, aufgeteilt. Die Zeichenfolge wird an jeder Stelle, an der ‹test_sexp› auftritt aufgeteilt. Die ‹test_sexp› Markierungen werden aus den Ergebnis-Strings entfernt sein. Das ‹result_Array$[]› wird ohne einen Index angegeben. Das Array muss vorher nicht dimensioniert worden sein.
Beispiel:

```
string$ = ”a:b:c:d”
argument$ = ”:”
split result$[], string$, argument$

array.length length, result$[]
for i = 1 to length
print result$[i] +  ;
next i
Print
```

Gibt aus: a b c d
Hinweis: Die ‹test_sexp› ist eigentlich ein regulärer Ausdruck. Wenn Sie nicht die Ergebnisse erreichen, die Sie mit ‹test_sexp› erwarten, dann sollten Sie die Regeln für reguläre Ausdrücke unter:
http://developer.android.com/reference/java/util/regex/Pattern.html prüfen.

Time Year$, Month$, Day$, Hour$, Minute$, Second$

Gibt die aktuelle Uhrzeit in den gewünschten String-Variablen zurück.

Tone <frequency_nexp>, <duration_nexp>

Spielt einen Ton der angegebenen Frequenz in Hertz für die angegebene Dauer in Millisekunden. Die Dauer die erzeugt wird, ist nicht exakt die angegebene Dauer. Wenn Sie diese brauchen, müssen Sie mit einem Experiment die exakte Dauer bestimmen. Jedes Android-Gerät hat eine minimale Ton- Dauer. Wenn Sie eine minimale Dauer angeben die kleiner ist als diese, erhalten Sie eine Laufzeit-Fehlermeldung die das Minimum für dieses Gerät angibt.

Vibrate <Pattern_Array[]>,<nexp>

Der Vibrationsalarm Befehl bewirkt, dass das Gerät in der angegebenen, ‹Pattern_Array[]›, Muster vibriert.
Das Muster ist von der Form: Pause, an, .., Pause, an. Das Muster kann eine beliebige Länge haben. Es muss mindestens zwei Werte haben, um ein Summen zu bekommen,

weil der erste Parameter eine Pause ist.
Die Werte für Pause und Anschalten sind Laufzeiten in Millisekunden.
Wenn ‹nexp› = -1 dann das Muster wird einmal abgespielt und nicht wiederholt werden.
Wenn ‹nexp› = 0, dann wird das Muster auch weiterhin gespielt, immer und immer wieder, bis das Programm beendet wird oder
wenn mit ‹nexp› › 0, das Muster- Abspielen abgebrochen wird.
Sehen Sie das Beispielprogramm, f21_sos.bas, für ein Beispiel des Vibrierens.

WakeLock <code_nexp>

Die WakeLock Befehl ändert das System Bildschirm Timeout-Funktion. Der ‹code_nexp› kann einer der folgenden fünf Werte sein. Die ersten vier Werte ändern Sie die Display-Timeout auf verschiedene Weise.

Code	CPU	Bildschirm	Tastatur Licht
1	On*	Off	Off
2	On	Dim	Off
3	On	Bright	Off
4	On	Bright	Bright

** Wenn Sie eine teilweise WakeLock haben, wird die CPU auch weiterhin laufen, unabhängig von irgendwelchen Zeiten und auch nach dem Drücken der Power-Taste durch den Benutzer. In allen anderen WakeLocks, wird die CPU laufen, aber der Benutzer kann immer noch das Gerät zum Schlafen einstellen, über den Netzschalter erreichen.*

Die fünfte Codewert, 5, löst den WakeLock und stellt die System-Screen Timeout-Funktion. Die WakeLock wird immer freigegeben, wenn das Programm nicht mehr läuft. Eine der häufigsten Anwendungen für WakeLock wäre in einem Musik-Player, der weiter Musik spielen muss, auch nach dem System-Screen Timeout-Intervall. Dieser Einsatz erfordert, dass BASIC! in Betrieb gehalten werden muss. Ein Weg, dies zu tun ist, BASIC! in eine Endlosschleife zu setzen:

```
Audio.load n,"B5b.mp3"
Audio.play n
WakeLock 1
Do
    Pause 30000
Until 0
```

Der Bildschirm wird ausgeschaltet, wenn die System Bildschirm Timeout-Zeit abgelaufen ist, aber die Musik wird weiterhin gespielt.

http.post url$, list, result
Führt einen Post- Befehl an einem Internet-Standort aus.
url$ ist ein String-Ausdruck der die URL (”http://.”.), als Post akzeptiert.
List ist ein Zeiger auf einen String-List, der die Namen/Wert-Paare die für die Post benötigt werden enthält.
Result$ ist eine String-Variable, in welche die Post Antwort gestellt wird.

myPhoneNumber <svar>
Die Telefonnummer des Android-Gerät wird in der String-Variable zurückgegeben. Wenn die Vorrichtung nicht mit einem Mobilfunknetz verbunden ist, wird der zurückgegebene Wert mehrdeutig sein.

Phone.call <sexpr>
Die Telefonnummer die in dem String-Ausdruck enthalten ist, wird aufgerufen. Sie müssen das Gerät angeschlossen haben um im Mobilfunknetz zu telefonieren.

Phone.RCV.init
Bereiten das Telefon vor einen Anruf mit phone.rcv.next zu erkennen.

Phone.RCV.Next <state_nvar>, <number_svar>
Der Zustand des Telefons wird als numerischer Wert des Zustands zurückgegeben. Eine Telefonnummer kann als String-Variable zurückgegeben werden.
Status = 0. Das Telefon ist im Leerlauf. Die Telefonnummer wird eine leere Zeichenfolge sein.
Status = 1. Das Telefon klingelt. Die Telefonnummer wird in der Zeichenfolge sein.
Status = 2. Das Telefon ist ausgehängt. Wenn es keine Telefonnummer (eine leere Zeichenfolge) angibt, dann wird ein herausgehender Anruf gemacht. Wenn es eine Telefonnummer angibt, dann wird ein eingehenden Anrufs ausgeführt.

Status 1 und 2 werden kontinuierlich so wie lange das Telefon klingelt oder das Telefon ausgehängt (abgenommen) bleibt, gemeldet.

Sms.send <number_sexp>, <message_sexp>
Die SMS-Nachricht in dem String-Ausdruck wird auf die Nummer in der String-Ausdruck geschickt. Dieser Befehl liefert keine Rückmeldung über den Versand der Nachricht. Das Gerät muss an ein Mobilfunknetz angeschlossen sein, um die SMS-Nachricht zu senden.

SMS.RCV.init
Bereitet das Abfangen einer empfangenen SMS mit dem SMS.NEXT Befehl vor.

SMS.RCV.Next <svar>

Liest die nächste empfangene SMS-Nachricht aus den empfangenen SMS Nachrichten in die String-Variable.

Die zurückgegebene Zeichenfolge enthält ”@” Wenn keine SMS-Nachricht in der Warteschlange vorhanden ist.

Der ”sms.rcv.init” Befehl muss ausgeführt werden, bevor der erste Befehl ”sms.rcv.next” aufgerufen wird.

Beispiel:

```
DO
DO % Loop until SMS received
  PAUSE 5000 % Sleep of 5 seconds
  SMS.RCV.NEXT m$ % Try to get a new message
UNTIL m$ <> "@" % "@" indicates no new message
PRINT m$ % Print the new message
UNTIL 0   % Loop forever
```

Email.send <recipient_sxep>, <subject_sexp>, <body_sexp>

Die E-Mail-Nachricht im Body String-Ausdruck wird an den benannten Empfänger mit dem Namen und Betreffzeile gesendet werden.

Headset <state_nvar>, <type_svar>, <mic_nvar>

Satus: 1.0 wenn Headset eingesteckt ist, -1 wenn kein Headset eingesteckt ist.

Typ: Ein Text beschreibt den Gerätetyp.

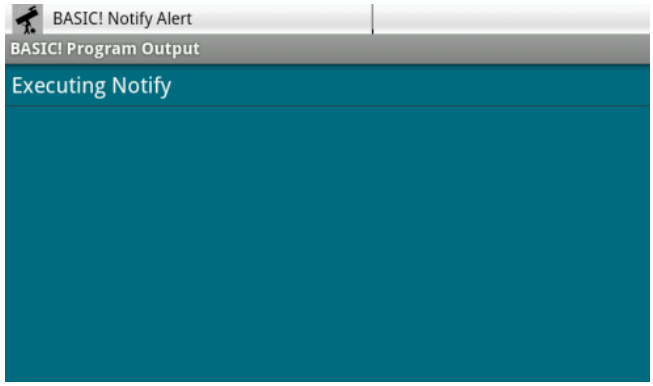
mic: 1,0, wenn das Headset über ein Mikrofon verfügt . -1 - wenn das Headset nicht über Mikrofon verfügt.

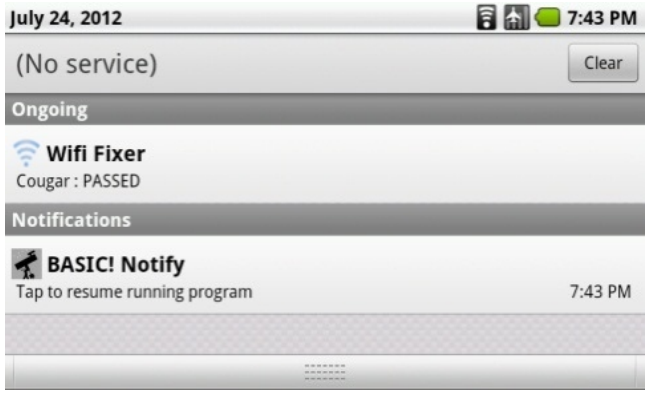
Notify <Title_sexp>, <Subtitle_sexp>, <alert_sexp>, <wait_nexp>

Dieser Befehl bewirkt, dass ein Notify-Objekt in der Notify (Status) Bar (Benachrichtigungfeld) platziert wird.

Wenn der numerische Ausdruck der Wartezeit nicht NULL, dann wird die Ausführung des BASIC! Programms angehalten, bis der Benutzer das Benachrichtigungsfeld anclickt. Wenn der Wert 0 (null) ist, wird BASIC! Programm weiter ausgeführt werden. Das Benachrichtigungsfeld (Notify-Objekt) wird entfernt, wenn der Benutzer das Objekt anclickt.

```
Print “"Executing Notify"”
        Notify “"BASIC! Notify”", "“Tap to resume running program”","”BASIC! Notify Alert"”, 1
        ! Execution is suspended and waiting for user to tap the Notify Object
        Print "“Notified”"
```





Hinweis: Das Symbol des Notify-Objekts wird das Symbol für das Apk sein, dass beim Erstellen verwendet wurde.

HOME (Startseite)

Der Startseite-Befehl tut genau das, was der Home-Taste tun würde. Der Home-Bildschirm wird angezeigt, während das BASIC! Programm weiterhin im Hintergrund ausgeführt wird.

OnBackGround:

Wenn der Hintergrund/Vordergrund- Zustand verändert wird und dieses Label im Programm anwesend ist, dann wird das aktuell ausgeführte Programm unterbrochen wird und der dem Label folgende Code ausgeführt. Die Background()-Funktion sollte verwendet werden, um den neuen Zustand bestimmen. Der unterbrochene Programmcode sollte durch Ausführen von Background.Resume fortgesetzt werden.

Background.Resume

Setzt die Ausführung des Programms nach einer Unterbrechung durch eine Zustandsänderung des Hintergrund/Vordergrund fort.

SQLITE

Überblick

Das Android-Betriebssystem bietet die Möglichkeit zum Erstellen, Pflegen und Zugreifen auf SQLite-Datenbanken. SQLite stellt eine in sich geschlossene, serverless, Zero-Konfiguration, Transaktions-SQL-Datenbank-Engine zur Verfügung.
SQLite ist die am häufigsten eingesetzte SQL-Datenbank-Engine der Welt. Mehr Einzelheiten über SQLite sind unter der SQLite-Homepage (http://www.sqlite.org/) zu finden.
Ein hervorragendes Online-Lernprogramm für SQL finden Sie unter www.w3schools.com. (http://www.w3schools.com/sql/default.asp).
Datenbank-Dateien werden auf der SD-Karte in dem Verzeichnis "/sdcard/rfo-basic/databases/" erstellt.

SQLITE Befehle

sql.open DB_Pointer, DB_Name$

Erstellt eine Datenbank für den Zugriff. Wenn die Datenbank nicht existiert, wird sie erstellt. DB_Pointer: Ist ein Zeiger auf die neu geöffnete Datenbank. Dieser Wert wird durch den sql.open Befehl gesetzt. DB_Pointer wird in den nachfolgenden Datenbank-Befehlen verwendet. DB_Pointer sollte nicht verändert werden. DB_Name$: Der Dateiname wird verwendet, um diese Datenbank zu erhalten. Das Basisreferenz Verzeichnis ist "data/data/com.rfo.basic/databases/". Wenn DB_Name$ = ": memory:" ist, wird eine temporäre Datenbank im memory (Arbeitsspeicher) erstellt.
Hinweis: Sie müssen möglicherweise mehr als eine Datenbank/Tabelle zur gleichen Zeit öffnen. Jedes geöffnete Datenbank/Tabelle muss seine eigene, besondere DB_Pointer Variable haben.

sql.close DB_Pointer

Schließt eine zuvor geöffnete Datenbank. Der DB_Pointer Wert wird auf Null zurückgesetzt. Die Variable kann dann in einem anderen sql.open.command wiederverwendet werden. Sie sollten immer eine geöffnete Datenbank schließen wenn Sie mit ihr fertig sind. Das Nicht-Schließen einer Datenbank kann die Größe des verfügbaren Speichers auf Ihrem Android Gerät sehr reduzieren.

sql.new_table DB_Pointer, DB_Name$, Table_Name$, C1$, C2$..,CN$

Eine einzige Datenbank enthält viele Tabellen. Eine Tabelle besteht aus Zeilen (Datenreihen), die aus einzelnen Daten bestehen. Eine Zeile/Datenreihe aus Daten ergibt Spalten der Werte. Jedem Wert der Spalte ist ein Spalten Name zugeordnet.

Der genannte Befehl erstellt eine neue Tabelle mit dem Table_Name$ in der referenzierten geöffneten Datenbank.
Die Spaltennamen für diese Tabelle werden durch die folgende Definition bestimmt: C1$, C2$, .., CN$: sind die Namen der Spalten. Mindestens eine Spalte Name ist erforderlich. Sie können beliebig viele Spaltennamen, wie Sie diese benötigen erstellen. BASIC! fügt immer eine Zeile Index Zeilennamen "_id" auf jede Tabelle. Der Wert in dieser Reihenindex- Spalte wird automatisch für jede neue eingefügte Zeile erhöht. Das gibt jeder Zeile in der Tabelle eine eindeutige Kennung. Diese Kennung kann verwendet werden, um Informationen in einer Tabelle mit einer anderen Tabelle zu verknüpfen. Für Beispielsweise kann der _id Wert für die Kundeninformationen in einem Kunden- Tabelle verwendet werden, um sie mit bestimmten Aufträgen von bestimmte Kunden mit einer spezifischen Ordnung der Datenbank zu verknüpfen.

sql.drop_table DB_Pointer, Table_Name$

Die Tabelle mit Table_Name$ in der geöffneten Datenbank, gekennzeichnet durch den DB_Pointer wird fallengelassen (dropped) wenn die Tabelle vorhanden ist.

sql.insert DB_Pointer, Table_Name$, C1$, V1$, C2$, V2$,,CN$, VN$

Fügt eine neue Zeile von Datenspalten und Werten in die zuvor geöffnete Datenbank. Table_Name$ ist der Name der Tabelle, in die die Daten eingefügt werden. Alle neu eingefügten Zeilen werden nach der letzten, bestehende Zeile der Tabelle eingefügt. C1$,V1$, C2$, V2$,CN$, VN$: sind die Spaltennamen-Wert-Paare für die neue Zeile. Diese Parameter müssen in Paaren sein. Für die Spaltennamen müssen die Spaltennamen verwendet werden, mit denen die Tabelle erstellt wurde. Beachten Sie, dass alle Werte Paare sind. Wenn Sie einen numerischen Wert für eine Spalte benötigen, verwenden Sie den BASIC! STR$(n) Befehl um die Zahl in einen String umzuwandeln. Sie können auch den BASIC! Format$ (pattern$, N), um eine formatierte Zahl für einen Wert zu erstellen. (Die Werte-als-Strings Voraussetzung ist ein BASIC! SQL-Schnittstellen Voraussetzung, nicht eine SQLite-Anforderung. Während SQLite, selbst alle Werte als Strings speichert, bietet es transparente Konvertierungen in andere Datentypen. Ich habe mich entschieden, nicht den Schnittstellenzugang zu diesen SQLite Umwandlungen zu erschweren und deshalb verfügt BASIC! über eigene Conversion-Fähigkeiten.)

sql.query Cursor, DB_Pointer, Table_Name$, Columns$, Where$, Order$

Fragt die geöffnete Datenbank-Tabelle nach einigen spezifischen Daten. Der Befehl liefert einen Cursor, wenn das Schreiten durch den Abfrage-Ergebnisse verwendet werden soll. Columns$: Ein String mit der Liste die Namen der Spalten wird

zurückgegeben. Die Spaltennamen sind durch Kommas getrennt. Ein Beispiel ist: Columns$ = ”Vorname,Nachname,Geschlecht,Alter” Wenn Sie die automatische Erhöhung des Row Index-Spalte erhalten möchten, dann gehören die ”_id” Spaltennamen in Ihrer Spalte zu ihrer Liste. Die Spalten können in beliebiger Reihenfolge aufgeführt werden. Die Reihenfolge der Spalten die in der Abfrage verwendet wird, wird angeben, wie die Daten der Zeilen zurückgegeben werden.
Where$: Der SQL-Zeichenfolgenausdruck wird verwendet, um auszuwählen, welche Zeilen zurückgegeben werden. Im Allgemeinen ist es ein SQL-Ausdruck von der Form ‹Spaltenname› ‹Operator› ‹Wert›. Zum Beispiel, Where$ = ”Vorname = ’John’”
Hinweis: Der Wert muss in einfache Anführungszeichen eingefügt sein. Genaue Informationen zu den SQL-Ausdrücke gibt es unter: http://www.sqlite.org/lang_expr.html Where$ ist ein optionaler Parameter. Wenn er weggelassen wird, werden alle Zeilen zurückgegeben werden. Der Order$ Parameter gibt die Reihenfolge an, in der die Zeilen zurückgegeben werden soll. Der Order$ Parameter wählt die Spalte, auf der die Ausgabezeilen sortiert werden. Er bestimmt auch, ob die Zeilen in aufsteigender (ASC) oder absteigend (DESC) Reihenfolge sortiert werden. Zum Beispiel, Order$ = ”Nachname ASC” würde die Zeilen nach dem Nachnamen sortiert von A bis Z zurückgeben. Wenn der Order$ Parameter vorhanden ist, muss der Parameter Where$ auch vorhanden sein. Wenn Sie wollen, daß alle Zeilen zurückgegeben werden, setzen Sie einfach Where$ = ””

sql.next Done, Cursor, C1V$, C2V$, .., CNV$

Wird der Cursor bei einen vorherigen Abfrage-Befehl benutzt, wird im nächsten Schritt die nächste Zeile von Daten generiert und auf die die Frage zurückgegeben.
C1V$, C2V$, CNV$ sind die Werte die zusammen mit den Spalten in dem Frage Columns$ String aufgelistet werden. Sie müssen die genaue Anzahl der Spaltenwerte in dieser Erklärung fordern, wie Sie diese in der Frage angefordert haben.
Wenn einer Ihrer Werte ein numerischer Wert ist, den Sie in arithmetischen Operationen benötigen, können Sie die BASIC! VAL(str$)-Funktion nutzen, um den Wert in eine Zahl umzuwandeln. Der Parameter Done/Fertig ist ein boolescher und wird verwendet, um zu signalisieren, dass die letzte Zeile der Abfrage eine Zeile zum Lesen zurückgegeben hat Wenn die letzte zurückgegebene Zeile gelesen wurde, ändert sie sich von 0 (falsch) auf 1 (true) Fertig. Wenn Fertig wahr wird, wird die Cursor-Variable auf Null zurückgesetzt. Sie kann nicht mehr für sql.next Operationen verwendet werden. Sie kann in einer nachfolgenden sql.query Anweisung verwendet werden. Sie können mehrere Cursor’s gleichzeitig geöffnet haben. Jeder geöffnete Cursor müßte natürlich einen

unterschiedliche Variablennamen haben.

sql.delete DB_Pointer, Table_Name$, Where$

Aus einer zuvor geöffneten Datenbank-Tabelle, werden Zeilen zu den Bedingungen von Where$ ausgewählt und gelöscht. Die Bildung/Formation des Where$ string ist genau die gleiche wie bei dem sql.query Befehl beschrieben.

sql.update DB_Pointer, Table_Name$, C1$, V1$, C2$, V2$,,CN$, VN$: Where$

In einem zuvor geöffneten Datenbank-Tabelle, werden Spaltenwerte in bestimmten Zeilen, die durch Where$ ausgewählt wurden, in Parameter geändert. Die C$, V$ Parameter müssen in Paaren angegeben sein. Der Doppelpunkt beendet die C$, V$-Liste und muss vor dem Where$ in diesem Befehl erscheinen.

sql.exec DB_Pointer, Command$

Führt ALLE nicht-Abfrage SQL-Befehle (CREATE TABLE, DELETE, INSERT, etc.) Zeichenfolge unter Verwendung einer zuvor geöffneten Datenbank-Tabelle aus.

sql.raw_query Cursor, DB_Pointer, Query$

Führt einen beliebigen SQL-Query-Befehl mit einer zuvor geöffneten Datenbank-Tabelle aus und und gibt einen Cursor für die Ergebnisse zurück.

Grafik

Einführung

Das Graphik-Display und der Grafik-Modus

Grafiken werden auf einem neuen Bildschirm, der anders als der BASIC!- Textausgabe Screen ist, angezeigt. Der Text Ausgabebildschirm ist noch vorhanden und kann auch noch beschrieben werden. Sie können aus dem Grafik-Bildschirm mit Hilfe der Zurück-Taste zurückgehen oder durch das Programm den Befehl gr.front ausführen.
Der gr.open Befehl öffnet die Grafik-Bildschirm und versetzt BASIC! in den Grafik-Modus. Basic muss im Grafik-Modus sein, bevor andere Grafik-Befehle ausgeführt werden können. Der Versuch, einen Grafik- Befehl auszuführen, wenn BASIC! nicht im Grafik-Modus ist, wird immer zu einem Laufzeitfehler führen. Der Graphics-Modus schaltet sich automatisch aus, wenn die BACK-Taste oder die MENU-Taste gedrückt wird. BASIC! wird auch weiterhin im Grafik-Modus laufen nachdem die BACK-Taste oder die MENU-Taste gedrückt wurde und dadurch die Output-Konsole wird angezeigt wird. Die BASIC! Output-Konsole wird ausgeblendet, wenn der Grafik-Bildschirm angezeigt

wird. Eine - Laufzeitfehler- Error Nachricht wird dadurch nicht zu sehen sein. Nur eine haptische Alarmsignal-Rückmeldung signalisiert einen Laufzeitfehler. Dieses haptische Feedback ist ein deutliches, kurzes Summen. Drücken Sie die Zurück-Taste, um den Grafik-Bildschirm nach dem Wahrnehmen dieser Warnung zu schließen. Die Fehlermeldungen können Sie dann auf der BASIC! Output-Konsole lesen.
Die gr.front Befehl kann verwendet werden, um beim vordersten Bildschirm zwischen der Output-Konsole und dem Grafik-Bildschirm zu wechseln. Wenn Ihr Programm endet, wird die Grafik-Bildschirm geschlossen werden. Wenn Sie den Grafik-Bildschirm als Anzeige erhalten wollen, wenn das Programm endet, setzen Sie eine Endlos-Schleife an das Programmende, wie z.B.:

```
! Grafik Bildschirm sehen
do
until 0
```

Display Lists

BASIC! zeigt eine Liste an. Die Anzeigeliste enthält Verweise/die Pointer auf alle Grafik-Objekte (Kreise, Linien, etc.) die befohlen wurden zu zeichnen. Die Objekte werden auf dem Bildschirm in der Reihenfolge, in der diese gezeichnet wurden angezeigt. Die Display-List-Objekte werden nicht auf dem Bildschirm dargestellt werden, bis der ”gr.render”-Befehl aufgerufen wurde. Jeder Zeichenobjekt-Befehl gibt für das Objekt eine Objekt Nummer in der Objektliste. Diese Objekt Nummer kann verwendet werden, um das Objekt schnell zu verändern. Sie können alle Parameter eines beliebigen Objekts in der Objektliste mit dem gr.modify Befehl verändern. Mit dieser Funktion können Sie einfache Animationen erzeugen, ohne den Aufwand der Neuerstellung des Objektes in der Objektliste. Jedes Mal, wenn ein Objekt in die Objekt-Liste hinzugefügt wird, wird die Objektnummer auch in die ursprüngliche Anzeige-Liste hinzugefügt. Der Anwender kann optional den gr.NewDL Befehl nutzen, um die anfängliche Anzeige mit einer benutzerdefinierten Display-Liste ersetzen.
Dieses Benutzer- Anzeigelisten Array kann einige oder alle der Objekt-Nummern der Objektliste enthalten.
Der primäre Einsatz von benutzerspezifischen Display-Listen ist, die Z-Reihenfolge der Objekte zu ändern. Mit anderen Worten: Sie können mit dieser Funktion ändern, welche Gegenstände auf andere Objekte gezogen werden. Sehen Sie sich dazu das Beispielprogramm, f24_newdl, als funktionierendes Beispiel für den gr.NewDL Befehl.

Zeichnen in Bitmaps

Sie können in Bitmaps zeichnen, aber auch zusätzlich dazu direkt auf dem Bildschirm zeichnen. Sie benachrichtigen BASIC! mit dem Befehl gr.bitmap.drawinto.start., wenn Sie das Zeichnen in ein Bitmap starten wollen, anstelle des Zeichnens direkt auf den Bildschirm Das versetzt BASIC! in den draw-into-bitmap-Modus. Alle Zeichenbefehle die in diesem Modus ausgegeben werden, zeichnen direkt in das Bitmap. Die Objekte die in diesem Modus gezeichnet werden, werden nicht in die Display-Liste aufgenommen. Die Objekt Nummer der zurückgegebenen draw-Befehle werden in diesem Modus als ungültig betrachtet und sollten nicht, insbesondere nicht für gr.modify verwendet werden.
Der draw-into-bitmap-Modus wird beendet, wenn der gr.bitmap.drawinto.end Befehl ausgegeben wird. Nachfolgende draw- Befehle werden als Objekte in die Anzeigeliste eingefügt, für das Rendern auf dem Bildschirm. Wenn Sie die Anzeige des drawn-into bitmap auf dem Bildschirm wünschen, führen Sie einen bitmap.draw Befehl für dieses Bitmap aus.

Farben

Android Farben bestehen aus einer Mischung von Rot, Grün und Blau. Jede der Farben können Werte im Bereich 0-255 haben. Schwarz tritt auf, wenn alle drei Werte Null sind. Weiß tritt auf, wenn alle drei Werte 255 sind.
Rot/Solid Red wäre Rot/Red mit einem Wert von 255, während Blau und Grün Null sind zu erreichen.
Die Farben haben damit, was man einen Alpha-Kanal nennt. Der Alpha-Kanal beschreibt den Grad der Undurchsichtigkeit der Farbe.
Ein Alpha-Wert von 255 ist völlig undurchsichtig. Kein Objekt in einer beliebigen Farbe ist durch ein anderes Objekt mit einem Alpha-Wert von 255 zu sehen. Ein Alpha-Wert von Null wird das Objekt beim Rendern unsichtbar machen.

Grafik-Setup-Befehle

gr.open alpha, red, green, blue {, ShowStatusBar {, Orientation}}

Öffnet das Graphics-Screen und versetzt BASIC! in den Grafikmodus. Die Farbwerte bestimmen die Hintergrundfarbe des Grafik-Bildschirms. Die Statusleiste wird auf dem Grafik- Bildschirm angezeigt, wenn ShowStatusBar nicht Null ist. Die Orientierung beim Öffnen des Grafik-Bildschirms wird durch den Orientierungs-Wert bestimmt werden. Die Orientierung Werte sind die gleichen wie die Werte für die gr.orientation Befehl.
Hinweis: Die ShowStatusBar ist optional, jedoch eine ShowStatusBar Wert muss sein, damit der derzeitige Orientierungswert angegeben werden kann. Der Standard-Wert

Orientation, wenn er nicht angegeben wird, ist Null (Querformat).

gr.color alpha, red, green, blue, fill

Setzt die aktuelle Farbe zum Zeichnen von Objekten. Die aktuelle Farbe wird unabhängig vom Grafik-Objekte verwendet und damit gezeichnet, bis der nächste neue Farbe- Befehl ausgeführt wird. Der BASIC! Farb-Befehl hat als zusätzlichen Parameter, das Füllen/fill. Der Füllungs- Parameter wird wie folgt angewendet
bei Objekten wie Kreisen oder Rechtecken. Wenn fill = false (0) ist, dann wird jedes eingeschlossene Objekt unter Verwendung dieser Farbe
nur als Umriss des Objekts angezeigt. Wenn fill = true ist (nicht 0), dann wird das eingeschlossene Objekt auch mit der Farbe gefüllt werden.

gr.set.stroke <nexp>

Stellt die Linienbreite der Objekte, die nach diesem Befehl ausgegeben werden ein. Die ‹nexp› Wert muss › = 0 sein. Null produziert die dünnste Linie und ist der Standard-Breiten Wert.

gr.orientation <nexp>

Legt die Ausrichtung des Bildschirms fest. Querformat ist festgelegt, wenn der Grafik-Modus geöffnet wird. Dies kann mit diesem Befehl geändert werden.
-1 = Orientierung hängt von den Sensoren ab.
0 = Orientierung wird auf Querformat gezwungen.
1 = Orientierung wird auf Hochformat gezwungen.
Sie können Änderungen in der Orientierung durch das Lesen der Bildschirm Breite und Höhe mit dem Befehl gr.screen überwachen.

Gr.StatusBar.Show <nexp>

This command has been deprecated. To show the status bar on the graphics screen, use the optional fifth parameter in gr.open.
Dieser Befehl rendert und zeigt alle Objekte, die in einer aktuellen Display-Liste sind. Ein Pause- Befehl nach einem gr.render Befehl ist nicht notwendig. Der gr.render Befehl wird nicht verlassen, bis die Anzeige Liste vollständig angezeigt ist.

gr.screen Breite, Höhe

Gibt die Bildschirm- Breite und Höhe in den Variablen zurück. Dieser Befehl sollte ausgeführt werden nach (nicht vor) jedem gr.orientation Befehl. Das ist nützlich, weil der gr.orientation Befehl die Höhe und Breite tauscht.

gr.scale x_factor, y_factor

Skaliere alle Zeichenbefehle durch die x-und y Skalierungsfaktoren. Dieser Befehl wird

zur Verfügung gestellt, damit in einem Gerät selbstständig, durch Skalieren der Zeichnung die tatsächliche Größe des Bildschirms berücksichtigt wird, auf dem Ihr Programm ausgeführt wird. Zum Beispiel:

```
! Setzen der geräteunabhängigen Größe
di_height = 480
di_ width = 800
! Sie erhalten die tatsächliche Breite und Höhe
Gr.open 255, 255, 255, 255
Gr.orientation 0
gr.screen actual_w, actual_h
! Berechnung der Skalierungsfaktoren
scale_width = actual_w /di_width
scale_height = actual_h /di_height
! Setzen der Skalierung
gr.scale scale_width, scale_height
```

Nun, beginnt das Zeichnen entsprechend der di_height Höhe und di_ width Breite des eingestellten Grafikbildschirms. Die Zeichnungen werden so skaliert, dass das sie Gerät angepasst werden können, während
das Programm läuft.

gr.cls

Löscht auf dem Bildschirm alle Gegenstände und entsorgt die aktuelle Objekt/ und Anzeigeliste. Es wird eine neue Objekt und benutzerspezifische Display-Liste. Alle vorherigen gr.color oder gr.text{size I align I bold I strike I underline I skew} Einstellung werden zurückgesetzt. Alle bisher erstellten Objekte werden gelöscht und alle bisherigen Objekt-Referenzen werden für ungültig erklärt. Die ”gr.render”-Befehl muss aufgerufen werden, um den Bildschirm für den Benutzer leergeräumt sichtbar zu machen.

gr.close

Schließt den geöffneten Grafik-Modus. Das Programm läuft weiter. Die Grafik-Bildschirm wird entfernt. Die Textausgabe wird angezeigt.

gr.front flag

Der Befehl bewirkt, dass der Grafik-Bildschirm oder die Ausgabe-Konsole, zum vordersten Bildschirm wird. Wenn flag = 0, wird als vorderer Bildschirm die Textausgabekonsole durch den Benutzer zu sehen sein. Wenn Flag ‹› 0 wird der Grafik-

Bildschirm als vorderste Bildschirm von dem Benutzer zu sehen sein.
Dieser Befehl ist sehr nützlich, um Eingaben des Benutzers während des Grafik-Modus zu realisieren. Die Ausführung ”gr.front 0 ”bewirkt, dass der Textausgabe Bildschirm, durch den Anwender angesehen werden kann. Das INPUT-Kommando kann ausgeführt werden. Wenn das Eingangssignal zum Ausführen von ”gr.front 1” empfangen wird, ist der Grafikbildschirm bereits für den Benutzer zu sehen. (Alternativ kann die Text.Input Befehl verwendet, um Text eingegeben wird, während in der Grafik-Modus zu erhalten ohne gr.front zu verwenden.)
Hinweis: Wenn die Output-Konsole vorne ist, kann man noch zeichnen (aber nicht rendern) auf die Grafik-Bildschirm. Der Befehl ”gr.front 1” muss vor jeder gr.render ausgeführt werden. Print-Textausgabe-Befehle werden auch weiterhin auf der Text-Output-Konsole ausgegeben, während der Graphic-Bildschirm im Vordergrund ist.

gr.brightness <nexp>
Legt die Helligkeit des Bildschirm-Grafik fest. Der Wert des numerischen Ausdrucks sollte zwischen 0.01 (Dunkel) und 1.00 (hell) sein.

Graphische Object- Creations Befehle

gr.line Object_number, x1, y1, x2, y2
Erstellt und fügt ein Linien- Objekt in die Anzeigeliste. Die Linie beginnt am Startpunkt (x1, y1) und endet am Punkt (x2, y2). Object_number gibt in die Objekt Liste die Objekt-Nummer für diese Linie. Dieses Objekt ist nicht sichtbar, bis der GR.RENDER-Befehl aufgerufen wird. Die dünnsten horizontalen und vertikalen Linien werden mit ”gr.set.stroke 0” gezeichnet. Diese Linien sind zwei Pixel breit, wenn AntiAlias an (on) ist. Schalten Sie AntiAlias aus (off) um horizontale und vertikale Linien in Einzelpixelbreite zu zeichnen. Die Parameter für gr.modify gr.line sind: ”x1”, ”y1”, ”x2” und ”y2”

gr.rect Object_number, links,oben, rechts, unten
Erstellt und fügt ein Rechteck-Objekt in der Anzeigeliste. Das Rechteck wird entsprechend der Grenzen durch die Parameter angeordnet. Das Rechteck wird gefüllt oder nicht gefüllt sein, abhängig von dem ”gr.color fill” Parameter. Object_number gibt in die Objekt Liste, die Objekt-Nummer für dieses Rechteck. Das Objekt wird erst sichtbar, nachdem der ”gr.render”-Befehl aufgerufen wurde. Die gr.modify Parameter für gr.rect sind: ”links”, ”oben”, ”rechts” und ”unten”.

gr.oval Object_number, links, oben, rechts und unten
Erstellt und fügt ein oval geformtes Objekt in die Anzeigeliste. Das Oval wird innerhalb der Grenzen der Parameter dargestellt. Das Oval wird gefüllt oder nicht gefüllt, je nach

dem gr.color Füllungsparameter. Object_number gibt in die Objekt Liste die Objekt-Nummer für dieses Oval. Dieses Objekt wird erst sichtbar, wenn der ”gr.render” Befehl aufgerufen wurde. Die Parameter für gr.modify gr.oval sind: ”links”, ”oben”, ”rechts” und ”unten”.

gr.arc Object_number, left, top, right, bottom, start_angle, sweep_angle, fill_mode

Erstellt und fügt ein bogenförmiges Objekt in die Anzeigeliste. Der Bogen wird innerhalb der durch das Rechteck beschriebenen Parameter erstellt. Er wird mit dem angegebenen start_angle (Winkel) beginnen und im Uhrzeigersinn schrittweise die Schrittwinkelgrade (Vektoren) zeichnen. Wenn der gr.color Parameter fill, true (ungleich 0) ist, wird der Bogen gefüllt. Wenn der Bogenparameter fill_mode = false (0) ist, wird der Bogen nur zwischen seinen Endpunkten gezeichnet werden oder zwischen diesen gefüllt. Wenn der Bogen fill_mode true (nicht 0) ist, werden die Bogenenden zusätzlich mit der Mitte des Bogens verbunden werden. Ein Fill_mode=true Bogen ist dadurch ein Keil- oder Kuchen- förmiges Objekt. Object_number übergibt der Objekt Liste die Objekt-Nummer für diesen Bogens. Dieses Objekt ist nicht sichtbar, bis der gr.render-Befehl aufgerufen wurde. Die Parameter für gr.modify gr.arc sind: ”links”, ”oben”, ”rechts”, ”unten”, ”start_angle”, ”end_angle” und fill_mode. Der Wert für ”fill_mode” ist entweder falsch (0) oder true (nicht 0);

gr.circle Object_number, x, y, radius

Erstellt und fügt ein Kreis-Objekt in der Objektliste. Der Kreis wird mit dem angegebenen Radius um die angegebene Mittelpunkt mit den (x, y)-Koordinaten erstellt. Der Kreis wird gefüllt oder nicht gefüllt abhängig von dem gr.color fill Parameter. Der Befehl übergibt der Objekt Liste die Objekt-Nummer für diesen Kreis. Das Objekt ist nicht sichtbar, bis der ”gr.render”-Befehl aufgerufen wurde. Die ”gr.modify” Parameter für gr.circle sind ”x”, ”y” und ”radius”

gr.set.pixels Object_number, Pixel [] {x, y}

Fügt ein Array von Pixel-Punkte in der Liste Objekt. Das Pixel []-Array enthält Paare von x-und y-Koordinaten für jedes Pixel. Die Pixel []-Array kann beliebig groß sein, sollte aber eine gerade Anzahl von Elementen haben. Wenn ein optionales x, y Daten- Paar vorhanden ist, können die x und y Koordinatenwerte des Arrays an jeder beliebigen Stelle des Programms gesetzt werden. Dies bietet die Möglichkeit, das Pixel-Array über den Bildschirm bewegen. Die Standardwerte für ein x-, y-Paar sind 0,0. Negative Werte für die x-, y-Paare sind gültig. Die modify Parameter für diesen Befehl sind ”x” und ”y”. Zusätzlich zum Ändern über modify, können die einzelnen Elemente der Pixelmatrix

sofort direkt geändert werden. Zum Beispiel
Pixel [3] = 120
Pixel [4] = 200
bewirkt, dass der zweite Pixel mit x = 120, y = 200 beim nächsten Darstellungserzeugungsintervall dargestellt wird.

Gr.poly Object_number, List_pointer {x, y}

Zeichnet ein geschlossenes Polygon mit einer beliebigen Anzahl von Seiten.
Der List_pointer ist ein Zeiger auf eine Listen- Datenstruktur. Die Liste enthält x, y-Koordinaten Paare. Die ersten Koordinaten definieren den Punkt, an dem die Polygon Zeichnung zu beginnen hat. Jedes nachfolgenden Koordinatenpaar definiert eine Linie von den letzten Koordinaten bis zu diesem Koordinatenpaar. BASIC! wird automatisch die endgültige, Polygon-Schließungs- Linie von den letzten angegebenen Koordinaten auf das erste Koodinatenpaar ziehen. Damit wird sichergestellt, dass das Polygon geschlossen ist. Die minimale Anzahl der Koordinatenwerte sind sechs (drei Paare). Drei Paare definieren ein Dreieck.
Linienbreite, Linienfarbe, alpha und Fill- Parameter des Polygons werden durch vorhergegangene gr.color und gr.set.stroke bestimmt wie bei jedem anderen gezeichneten Objekt. Wenn ein optionales x, y Daten- Paar vorhanden ist, können die Werte der x und y Koodinaten der Liste an jeder beliebigen Stelle des Programms hinzugefügt werden. Dies bietet die Möglichkeit, das Polygon-Array über den Bildschirm bewegen. Die Standardwerte für das x-, y-Paar sind 0,0. Negative Werte für das x-, y-Paar sind gültig. Die ”gr.modify”- Parameter sind ”x”, ”y” und ”list”
Sehen Sie sich das Beispielprogramm, f30_poly, für ein funktionierenden Beispiel von gr.poly an.

Ein-und ausblenden Befehle (Hide and Show Commands)

gr.hide Object_number

Blendet das Objekt mit dem angegebenen Object_number aus. Diese Änderung wird erst sichtbar nachdem der gr.render Befehl aufgerufen wurde.

gr.show Object_number

Zeigt (Shows - unhiddes) das Objekt mit dem angegebenen Object_number. Diese Änderung wird nicht sichtbar sein, bis der gr.render-Befehl aufgerufen wurde.

Berührungs - Abfragebefehle - Touch Query Commands

Wenn Sie Bildschirmberührungen verwenden und Sie bewegen die Finger ohne

Anheben über den Bildschirm, kann die Bewegung verfolgt werden, indem Sie die Touch Query - Abfragebefehle wiederholt aufrufen. Dies ermöglicht Ihnen ein Ziehen von Grafikobjekte auf dem Bildschirm zu programmieren. Das Beispielprogramm, f23_breakout.bas, illustriert dies mit dem Code, der das Paddel bewegt.
Das onTouch: Label kann wahlweise Ihr Programm unterbrechen, wenn eine neue Berührung erkannt wurde.
Die Touch-Befehle berichten darüber, ob einer oder zwei Finger den Bildschirm berühren. Wenn zwei Finger sich kreuzen, einer auf der y- der andere auf der x-Achse, dann wird die Berührung mit der Berührung-2 getauscht.

gr.touch touched, x, y

Abfrage nach einer Berührung des Grafik- Bildschirms. Wenn der Bildschirm berührt wurde, wird touched als wahr (true) mit
(Nicht 0) und mit der (x, y)-Koordinaten der Berührung zurückgegeben. Wenn berührt false (0) ist, dann wurde der Bildschirm nicht
berührt und die (x, y) Koordinaten sind ungültig. Der Befehl wird das weiterhin zurückgeben, bis der Bildschirm berührt wurde.
Wenn Sie ein einzelnes, kurzes Antippen, nach der Erkennung der Berührung erkennen wollen, wäre eine until Schleife, bis Berührung (until touched) falsch.

```
do
gr.touch touched, x,y
until touched

// Touch erkannt, warten auf
// Finger Hebung
do
gr.touch touched, x, y
until
! touched- berührt
```

Die zurückgegebenen Werte beziehen sich auf die eigentliche Bildschirmgröße. Wenn Sie den Bildschirm skaliert haben, müssen Sie die zurückgegebenen Parameter entsprechend skalieren. Wenn die Parameter, die in gr.scale verwendeten wurden Scale_x und Scale_y (gr.scale Scale_x, Scale_y) waren, dann multiplizieren Sie die zurückgegebenen x und y durch die gleichen Werte.

```
gr.touch touched, x, y
Xscaled = X * Scale_x
```

Yscaled = y * Scale_y

gr.bounded.touch2 touched, links, oben, rechts und unten

Der touched-berührt Parameter wird als true (nicht 0) zurückgegeben, wenn der Benutzer den Bildschirm innerhalb des Rechteck das die links, oben, rechts, unten Parameter definiert ist. Wenn der Bildschirm wurde nicht berührt wurde oder hat außerhalb des umschließenden Rechtecks berührt wurde, wird der Parameter touched false (Null) sein. Das Kommando wird kontinuierlich als true (nicht 0) zurückgegeben, solange der Bildschirm berührt bleibt.
Die Grenzen im Feld Parameter ist die eigentliche Bildschirmgröße. Wenn Sie den Bildschirm skaliert haben, müssen Sie auch das Feld Begrenzungsparameter entsprechend skalieren. Wenn Sie die Parameter, die in gr.scale - Scale_x und Scale_y (gr.scale Scale_x, Scale_y) verwendet waren skaliert haben, dann multiplizieren Sie diese mit links und rechts von Scale_x und oben und unten von Scale_y.

gr.touch2 touched, x, y

Das gleiche Kommando wie gr.touch, außer dass es über eine zweite gleichzeitige Berührung des Bildschirms berichtet

OnTouch:

Wenn dieses Label im Programm vorhanden ist, wird der aktuell laufende Programmcode unterbrochen werden, wenn der Benutzer den Bildschirm berührt. Die Programmausführung wird dann die Anweisungen nach dieser Bezeichnung ausführen. Einer der oben genannten gr.touch Befehle kann ausgeführt werden.

gr.onTouch.Resume

Setzt die Ausführung des Programms an der Stelle, wo die Berührungsunterbrechung aufgetreten ist.

Textkommandos- Text Commands

gr.text.align type

Platzieren Sie den Text relativ zu den (x, y)-Koordinaten, wenn sie mit dem ”gr.text.draw”-Befehl schreiben wollen.
type Werte: 1 = links, 2 = Mitte, 3 = Rechts.

gr.text.size n

Bestimmt die Größe des Textes in Pixeln
Die Größe entspricht der Höhe der Buchstaben über der Zeilenlinie. Einige Zeichen reichen auch unter das Zeilenlinie, so dass die Gesamthöhe eines Textbereichs etwa 30% größer ist.

gr.text.width <nvar>, <sexp>
Gibt die Breite in Bildpunkten von ‹sexp› in ‹nvar› zurück. Die Breite wird vom letzten gr.text.size Befehl bestimmt.

gr.get.textbounds <sexp>, left, top, right, bottom
Ruft das Begrenzungsrechteck des Zeichenfolgenausdruck auf. Der Ursprung des Rechtecks wird als 0,0 angenommen. Der Wert der für ”top” zurückgegeben wird, ist eine negative Zahl. Deshalb, weil die X-Koordinate durch den gr.text.draw-Befehl den untersten Teil des Textes spezifiziert. Wenn dies verwirrend ist, versuchen Sie dieses Beispiel ausführen:

```
gr.open 255,255,255,255
gr.color 255,255,0,0,0
gr.text.size 40
gr.text.align 1
s$ = "This is a test"
gr.get.textbounds s$,l,t,r,b
print l,t,r,b
gr.rect x,l+10,t+50,r+10,b+50
gr.text.draw x,10,50,s$
gr.render
pause 5000
```

gr.text.typeface type

Legt die Text-Schriftart und Text-Stil fest. Die Werte für type sind:

1 = Standard-Schriftart

2 = Monospace font

3 = San-Serif font

4 = Serif font

gr.text.bold Boolean
Schreibt den Text fett, wenn Boolean ‹› 0 ist. Wenn die gr.color fill Parameter 0 ist, werden nur die Umrisse des fetten Textes gezeigt. Wenn füllen ‹› 0, wird die Textaußenline gefüllt.

gr.text.skew n
Verzerrt den Text, um eine kursive Wirkung zu geben. Negative n Werte verschieben der Boden des Textes nach links. Dies bewirkt, daß der Text sich nach vorne beugt. Positive Werte bewirken das Gegenteil. Traditionelles Kursiv lässt sich am besten mit n = -0,25 imitieren.

gr.text.strike Boolean
Die gezeichneten Text wird unterstrichen dargestellt, wenn Boolean ‹› 0 ist.

gr.text.draw Object_number, x, y, text$

Erstellt und fügt ein Text-Objekt in der Objektliste. Das Text-Objekt wird mit der letzten Farbe und entsprechend den letzten text vorbereitenden Befehlen dargestellt. Fügt in die Objekt Liste die Objekt-Nummer für diesen Text ein. Dieses Objekt ist nicht sichtbar, bis der ”gr.render”-Befehl aufgerufen wurde.
Der y-Wert entspricht der niedrigsten Position jedes Zeichen in der Zeichenfolge.
Die ’Zeilen- Linie’ liegt in der Regel bei plus 30% der gr.text.size Größe.
Die ”gr.modify” Parameter für gr.text.draw sind ”x”, ”y”, ”text”. Der Wert für den ”text”-Parameter ist ein string, der den neuen Text repräsentiert.

Bitmap-Befehle- Bitmap Commands

Gr.bitmap.create bitmap_ptr, Breite, Höhe

Erstellt ein leeres Bitmap in der angegebenen Breite und Höhe. Die angegebene Breite und Höhe kann, wenn nötig größer als die Größe des Bildschirms sein.
Gibt einen Zeiger auf das erstellte Bitmap-Objekt für den Einsatz mit den anderen gr.bitmap Befehle zurück.

gr.bitmap.load bitmap_ptr, file_name $

Erstellt eine Bitmap-Objekt aus einer Datei die in dem File_name $ string angegeben ist. Gibt einen Zeiger auf das erstellte Bitmap-Objekt für den Einsatz mit den gr.bitmap Befehlen zurück. Bei Bitmap-Bild-Dateien wird davon ausgegangen, dass sie sich in dem ”/ SDCard / RFO-basic / data /” Verzeichnis befinden.
Hinweis: Sie können Pfad Bezeichnungen in den Dateinamen aufzunehmen. Zum Beispiel, ”.. /.. / Cougar.jpg” würde Basic! dazu bringen Cougar.jpg in der obersten Ebene der SD-Karte. (”/sdcard/Cougar.jpg”) zu suchen.
”images/Kitty.png” würde dazu führen, dass BASIC! in dem Unterverzeichnis images(d) (”/sdcard/rfo-basic/data/images/Kitty.png”) nachsieht.

gr.bitmap.size bitmap_ptr, Breite, Höhe

Gibt die Pixel-Breite und Höhe des Bitmap, mit Namen bitmap_ptr in den Variablen-Werten Breite und Höhe zurück.

gr.bitmap.scale dest_ptr, src_ptr, Breite, Höhe{, Glättung}

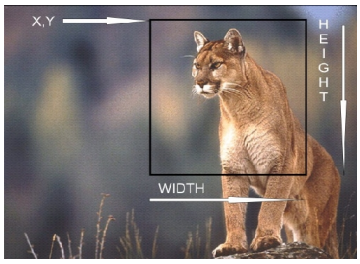
Skaliert ein zuvor geladenes Bitmap in die angegebene Breite und Höhe und erstellt eine neues Bitmap. Der src_ptr ist der Bitmap-Zeiger der von gr.bitmap.load oder einem anderen Bitmap-Erstellungs Befehl zurückgegeben wurde. Die dest_ptr ist der Bitmap-Zeiger für das neue, skalierte (verkleinerte oder vergrößerte) Bitmap. Minus-Werte für Breite und Höhe führen dazu, dass das Bild umgedreht wird, von rechts nach links oder

auf den Kopf. Weder die Breite noch der Höhe Wert darf Null sein. Die optionale Glättung als numerischer Ausdruck kann so verwendet werden, dass das skalierte Bild nicht geglättet wird. Wenn der Ausdruck Null ist, dann wird das Bild nicht geglättet werden. Wenn der optionale Parameter nicht angegeben, dann wird das Bild geglättet.

gr.bitmap.delete bitmap_ptr

Löscht ein zuvor geladenes oder skaliertes Bitmap. Der Bitmap-Speicher wird an das System zurückgegeben.

gr.bitmap.crop <new_bitmap_object_nvar>, <source_bitmap_object_nexp>, <x_nexp>,<y_nexp>, <width_nexp>, <height_nexp>



Das zuvor geladenen Quell- Bitmap durch ‹source_bitmap_object_nexp› vertreten, wird beschnitten. Der entstandene, neue Bitmap-Objekt-Pointer wird in neuer Form in ‹new_bitmap_object_nvar› zurückgegeben. Die ‹x_nexp›, gibt ‹nexp› das Punkt- Paar im Quell-Bitmap an, an dem das Beschneiden startet. Das ‹width_nexp› - ‹height_nexp› Paar gibt die Größe des verbleibenden Bitmaps an.

gr.bitmap.save Object_ptr, "Dateiname" {, <quality_nexp>}

Speichert die angegebene Bitmap in einer Datei. Der standardmäßige Pfad lautet ”/sdcard/rfo-basic/data/”

Die Datei wird als JPEG-Datei gespeichert, wenn der Dateiname mit ”. jpg” endet.

Die Datei wird als PNG-Datei gespeichert werden, wenn der Dateiname mit ”. png” endet. Die Datei wird auch als PNG-Datei gespeichert, wenn der Dateiname, kein ”. png” oder ”. jpg” am Ende hat.

gr.bitmap.draw Object_ptr, bitmap_ptr, x, y

Erstellt und fügt ein Bitmap-Objekt in die Objektliste ein. Das Bitmap wird mit seiner oberen linken Ecke an die vorgesehen (x, y)- Koordinaten gesetzt. Fügt in die Objektanzeigeliste die Nummer für dieses Bitmap. Das Objekt wird nicht sichtbar sein, bis der ”gr.render”-Befehl aufgerufen wurde. Der Alpha-Wert des letzten gr.color bestimmt die Transparenz des Bitmap.

Die ”gr.modify” Parameter für gr.bitmap.draw sind ”Bitmap”, ”x” und ”y”.

gr.get.bmpixel bitmap_ptr, x, y, Alpha-, Rot, Grün, Blau

Gibt die Farbe für das Pixel der genannten Bitmap an der angegebenen x, y-Koordinate. Die x und y Werte dürfen nicht überschritten werden.

gr.bitmap.drawinto.start Bitmap_Pointer

Versetzt BASIC! an den draw-into-bitmap Modus. Alle Zeichen- Befehle werden so erteilt, dass sie in diesem Modus direkt in die Bitmap zeichnen. Die Objekte die in diesem Modus gezeichnet wurden, werden nicht in der Display-Liste platziert. Die Objekt-Nummer wird durch die Auslösung dieses Befehles in diesem Modus als ungültig zurückgegeben betrachtet werden und sollte nicht für andere Zwecke verwendet werden, auch nicht für gr.modify.

gr.bitmap.drawinto.end

Beendet den Draw-in-Bitmap-Modus. Nachfolgende Zeichenbefehle werden als Objekte in der Anzeigeliste platziert, für die Darstellung auf dem Bildschirm. Wenn
Wenn Sie den draw-into-bitmap auf dem Bildschirm Modus wieder zum Anzeigen möchten, führen Sie den Befehl für bitmap.draw wieder neu aus. Die Breite und Höhe des Bildschirms darf nicht kleiner als Null sein.

Dreh- Befehle - Rotate Commands

gr.rotate.start Winkel, x, y {, <obj_nvar>}

Alle Objekte die zwischen dem gr.rotate.start und gr.rotate.end gezeichnet wurden, werden in einem bestimmten Winkel, um den angegebenen (x, y) Koordinatenpunkt. Auf gr.rotate.start muss ein gr.rotate.end folgen, oder die erwarteten Ergebnisse werden nicht erzielt.
Die optionale ‹obj_nvar› enthält die Objekt-Liste der Objekt-Nummern des gr.rotate.start. Wenn Sie gedrehte Objekte im Array für gr.NewDl verwenden wollen, dann müssen diese die gr.rotate.start und gr.rotate.end Objekte umfassen.

gr.rotate.end {<obj_nvar>}

Beendet das Zeichnen von Objekten die gedreht werden. Objekte die nach diesem Befehl erstellt werden, werden nicht mehr gedreht dargestellt. Die optionale ‹obj_nvar› enthält die Objekt-Liste mit Objekt-Nummern des gr.rotate.end. Wenn Sie gedrehte Objekte im Array für gr.NewDl verwenden wollen, dann müssen die Objekt zu den gr.rotate.start und gr.rotate.end Objekten gehören.

Kamera-Befehle

Es gibt drei Möglichkeiten, um die Kamera aus BASIC! heraus zu verwenden.
1) Das Gerät ist so gebaut, dass die Camera User Interface Schnittstelle verwendet werden kann, um ein Bild zu erfassen. Diese Methode bietet den Zugang zu allen Bildaufnahme Funktionen, die Sie erhalten, wenn Sie die Kamera-Anwendung des

Gerätes aufrufen. Der Unterschied ist, dass das Bitmap-Bild an BASIC! zurückzugeben ist, für die Manipulation durch BASIC! Der gr.camera.shoot Befehl implementiert diesen Modus.
2) Ein Bild kann automatisch übernommen werden, wenn der Befehl ausgeführt wird. Dieser Modus ermöglicht eine unbegrenzte zeitsequenzierte Bilderfassung. Der Befehl bietet die Festlegung von Blitz auf An, Aus und Auto. Der gr.camera.autoshoot Befehl implementiert diesen Modus.
3) Der dritte Modus ist der gr.camera.manualshoot Befehl welcher sehr ähnlich wie der autoshoot Modus ist. Der Unterschied ist, dass eine Live-Vorschau zur Verfügung gestellt wird und das Bild wird nicht erfasst, bis der Bildschirm berührt wird.
Alle Bilder werden mit voller Kameraauflösung gemacht und gespeichert in 100 % jpg-Qualität als "/ sdcard/rfo-basic/data/image.png"
Alle diese Befehle geben auch einen Zeiger auf Bitmaps zurück. Die hergestellten Bitmaps werden um den Faktor 4 reduziert. Sie können am Ende mehrere andere Bitmaps von diesen zurückgegeben Bitmaps machen. Zum Beispiel: Sie müssen oft die zurückgegeben Bitmap skalieren, um sie auf den Bildschirm einzupassen. Alle Bitmaps, die Sie nicht vorhaben darzustellen sollen mit gr.bitmap.delete gelöscht werden, damit eine out-of-memory Situation vermieden wird. Sie sollten deshalb Bitmaps löschen, wenn Sie diese nicht mehr zum Zeichen und Darstellen nutzen.
Das Beispiel-Programm, f33_camera.bas, zeigt alle Modi der Kamera erklärt. Außerdem enthält es Beispiele für die Skalierung des Bildes, um es auf den Bildschirm einzupassen, das Schreiben von Text auf das Bild und das Löschen überflüssiger Bitmaps.
Das Beispiel-Programm, f34_remote_camera.bas, zeigt die Remote-Bilderfassung zur Verwendung von zwei verschiedene Android-Geräten.

Gr.camera.select 1|2

Wählt die Rückseite (1) oder die Front(2) bei Geräten mit zwei Kameras. Die Standard-Kamera ist die Rückseiten-Kamera (gegenüber dem Bildschirm). Wenn die einzige Kamera eine Front-Kamera (wie in der Nexus-7) ist, dann wird sie die Standard-Kamera.

gr.camera.shoot bm_ptr

Der Befehl ruft das als Kamera-Benutzerschnittstelle gebaute Gerät auf, um ein Bild zu machen. Das Bild wird zurückgegeben an BASIC! als ein Bitmap, gekennzeichnet mit der numerischen Variable bm.ptr. Wenn die Kamera-Schnittstelle keine Grundlage hat, ein Bild aufzunehmen, wird bm_ptr mit einem Null-Wert zurückgegeben werden.
Viele der Kamera-Schnittstellen- Geräte speichern auch die aufgenommenen Bilder noch

in irgendwelchen andern Speichern mit einem das Datum codierenden Dateinamen. Diese Bilder können mit einer Galerie-Anwendung gefunden werden. BASIC! ist nicht in der Lage zu verhindern, dass diese Dateien erstellt werden.
Hinweis: Einige Geräte wie das Nexus 7 werden nicht mit einer vorinstallierten Kamera Schnittstelle geliefert. Wenn Sie eine Aftermarket-Kamera-Anwendung installiert haben, dann wird diese beim Ausführen dieses Befehls aufgerufen werden. Sie können Bilder mit dem Nexus 7 (oder ähnliche Geräten) mit den anderen Befehlen aufnehmen, auch wenn keine Kamera-Anwendung installiert ist.

gr.camera.autoShoot bm_ptr {, flash_ mode}

Ein Bild wird, wenn der Befehl ausgeführt wird, erfasst. Es ist kein weiterer Benutzereingriff erforderlich. Dieser Befehl kann für die unbegegrenzte, zeitliche Abfolge von Bildaufnahmen genutzt werden.
Der optionale numerischer Ausdruck flash_mode, spezifiziert die Blitzlicht-Operationen.
0 = Auto Blitz
1 = Blitz an
2 = Blitz aus
Der Standardwert, wenn kein Parameter angegeben wird, ist Auto Blitzlicht.
Der Befehl speichert auch das aufgenommene Bild in der Datei ”/sdcard l/rfo-basic/data/image.png”

Gr.camera.manualShoot bm_ptr {,flash_ mode}

Dieser Befehl ist ähnlich wie die gr.camera.autoshoot Aufnahme, aber es wird eine Live-Vorschau auf dem Bildschirm dargestellt. Das Bild wird nicht erfasst, bis der Benutzer den Bildschirm angetippt hat.

Verschiedene Befehle

gr.screen.to_bitmap bm_ptr

Der aktuelle Inhalt des Bildschirms wird in ein Bitmap platziert. Der Zeiger auf das Bitmap wird in der bm_ptr Variable zurückgegeben.

gr.get.pixel x, y, Alpha-, Rot, Grün, Blau

Gibt die Farbe für das Bildschirm-Pixel an der angegebenen x, y-Koordinate zurück. Die x-und y-Werte dürfen nicht die Breite und Höhe des Bildschirms überschreiten und dürfen nicht kleiner als Null sein.

gr.save "Dateiname" {, <quality_nexp>}

Speichert den aktuellen Bildschirm in eine Datei. Der standardmäßige Pfad lautet

”/sdcard /rfo-basic /data/” Die Datei wird als JPEG-Datei gespeichert werden, wenn der Dateiname mit ”. jpg” endet. Die Datei wird als PNG-Datei gespeichert werden, wenn der Dateiname mit ”. png” endet. Die Datei wird auch als PNG-Datei gespeichert werden, wenn der Dateiname kein ”. png” oder ”. jpg” am Ende hat.
Das optionale ‹quality_nexp› wird verwendet, um die Qualität der gespeicherten JPEG-Datei anzugeben. Der Wert kann im Bereich von 0 (sehr schlecht) bis 100 (sehr gut) sein. Der Standardwert ist 50. Der Qualität Parameter hat keine Auswirkungen auf und PNG-Dateien. PNG-Dateien werden immer auf höchstem Niveau gesichert.
Hinweis: Die Dateigröße der JPEG-Datei ist proportional zur Qualität. Niedrigere Qualität produziert kleinere Dateien.

Gr.Get.Position Object_number, x, y

Holt die aktuelle X, Y-Position des angegebenen Display-Listen Objekts. Wenn das Objekt angegeben wurde mit den Rechteck- Parametern (Links, Oben, Rechts, Unten), dann wird Links in x zurückgegeben und Oben wird in y zurückgegeben.

Gr.Get.Value Object_number, tag$, value

Durch tag$ (”Links”, ”Radius”, etc.) werden die in dem angegebenen Objekt dargestellten Werte werden im Wert zurückgegeben.

Gr.Get.Value Text_Object_number, "text", theText$

Der Text im angegebenen Text-Objekt wird in TheText$ zurückgegeben

gr.modify Object_number, parameter_name$, {value|value$}

Das Objekt in der Anzeigeliste mit der angegebenen Object_number wird in seinem Parameter parameter_name$, geändert in {value oder value$}. Die Parameter für die einzelnen Objekte sind in den Beschreibungen der Objekt- Befehle beschrieben.
Als ein Beispiel soll ein Bitmap-Objekt das mit ”gr.bitmap.draw BM_ptr, galaxy_ptr, 400,120”. erstellt wurde, dienen.
Die Ausführung ”gr.modify BM_Ptr ”x”, 420” würde das Bitmap von x = 400 nach x = 420 versetzen.
Die Ausführung ”gr.modify BM_ptr, ”y”,200” würde das Bitmap von y = 120 nach y = 200 bewegen.
Die Ausführung ”gr.modify BM_ptr,” bitmap”, Saturn_ptr würde das Bitmap, des Bildes einer (vorinstallierten) Galaxy zu dem Bild eines (vorinstallierten) Saturn ändern.
Die Parameter, die Sie für ein bestimmtes Objekt liefern, werden nicht auf Richtigkeit oder Richtigschreibung für den Parameter überprüft.
Der Parameter wird auch nicht auf die Angemessenheit für das angegebene Objekt überprüft. Wenn Sie sind nicht immer die erwarteten Ergebnisse erreichen, prüfen Sie

die Parameter darauf, ob sie für das Objekt angemessen sind und auf Richtigschreibung. Die Angabe unpassender Parameter für ein Objekt, wird keine Auswirkungen auf das Objekt haben.
Die Ergebnisse der gr.modify Befehle werden nicht angezeigt, bis der Befehl gr.render gegeben wurde.
Normalerweise bekommen Grafikobjekte ihre Alphakanalwerte (Transparenz) von dem letzten gr.color Befehl. Diese Alphakanalwerte können explizit durch gr.modify geändert werden, ohne den letzten Farbwert zu ändern. Dies kann verwendet werden, um Objekte langsam erscheinen und wieder verschwinden zu lassen.

```
Do
        For a = 1 to 255 step 10
                gr.modify object,alpha,i
                gr.render
                pause 250
        next a
        For a = 255 to 1 step -10
                gr.modify object,alpha,i
                gr.render
                pause 250
        next a
until
```

Das Einstellen des Alpha-Wert eines Objektes auf 256, bringt BASIC! dazu den Alpha-Wert aus dem neuesten Color-Wert zu verwenden.

gr.paint.get <object_nvar>

BASIC! hat Farbgebungs-(Paint)- Objekte vorgesehen. Jedem Befehl, der etwas auf den Bildschirm zeichnet, ist ein Paint-Objekt zugeordnet. Die Paint-Objekte enthalten Informationen die aus gr.color und gr.text Befehlen abgeleitet sind.
Jedesmal, wenn ein gr.color oder gr.text Befehl ausgeführt wird, wurde auch ein neues Paint-Objekt erstellt. Das neue Paint Objekt enthält die Informationen aus der letzten erstellten Farbgebung des Objektes als die aktuelle gr.color oder gr.text Befehle gesetzt oder geändert wurden. Das letzte erstellte Paint- Objekt ist das Paint- Objekt, daß einem draw Objekt zugeordnet wurde, als der draw Befehl (zum Zeichnen) ausgeführt wurde.
Die gr.paint.get Befehle nehmen den Objekt-Zeiger des letzten geschaffenen Paint-Objektes. Dieser Objekt Zeiger kann verwendet werden, um das Paint- Objekt zuzuordnen allen vorherigen draw-Objekten die durch den gr.modify Befehl geändert

wurden. Die gr.modify Parameter ist ”paint”.

gr_collision (<object_1_nvar>, <object_2_navr>)

Die Objekt ‹nvar› s sind Objekt- Tabellen Zahlen, die zurückgegeben werden, wenn die Objekte erstellt wurden.
Wenn die Grenzen der Boxen der beiden Objekte überlappen, dann wird die Funktion true (nicht Null) zurückgegeben, sobald sie das tun.
Wen sie nicht überlappen, dann wird die Funktion false (Null) zurückgegeben.
Objekte, die auf Kollision geprüft werden können, sind: Rechteck, Bitmaps, Kreis, Bogen und Elypsen. Im Falle eines Kreises, Bogen und einer Elypse wird für das Objekt die rechteckige Begrenzungsbox für die Prüfung der Kollision zum Einsatz kommen, nicht das tatsächliche gezeichnete Objekt.
Hinweis: gr_collision ist eine Funktion, kein Befehl.

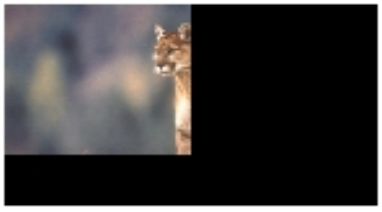
gr.clip <object_nvar>, <left_nexp>, <top_nexp>, <right_nexp>, <bottom_nexp> {, <RO_nexp>}

Clip Objekte die nach diesem Befehl gezeichnet werden, werden nur innerhalb der Grenzen des angegebenen gezogenen Cliprechtecks dargestellt.
Das Clip- Rechteck wird durch die numerischen Ausdrücke ”links, oben, rechts, unten” angegeben. Der letzte, optionale, Parameter ist der Region Operator, ‹RO_nexp›. Der Region Operator gibt vor, wie der Clip mit dem aktuellen Clip zu interagieren hat. Der Vollbildmodus ist der aktuelle Clip der vor dem ersten Clip ausgegeben wurde. Die RO-Werte sind:

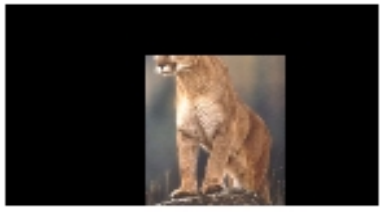
0 = Intersect (Schneiden)
1 = Difference (Unterschied)
2 = Replace (Ersetzen)
3 = Reverse Difference (Umgekehrte Differenz)
4 = Union (Vereinigung)
5 = XOR (Kontravalenz)

Beispiele:

Original

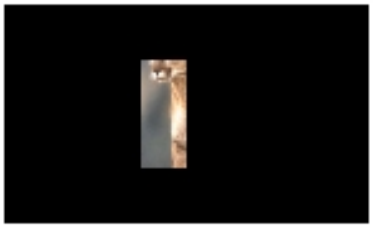


Clip 1

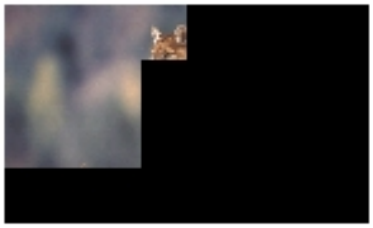


Clip 2

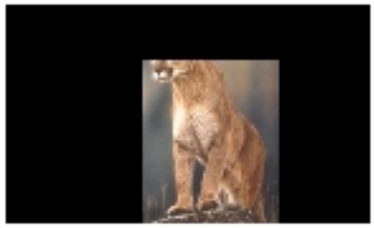
Clip 2 angewendet auf Clip 1 mit dem RO parameter von Clip 2



0 = Intersect (Schneiden)



1 = Difference



2 = Replace (Ersetzen)

3 = Reverse Difference

4 = Union (Vereinigung)

5 = XOR (Kontravalenz)

gr.clip ist ein Anzeigelistenobjekt. Es kann mit gr.modify modifiziert werden. Die Parameter die zu ändern sindsind ”links” ”oben” rechts unten RO
Die gr.show und gr.hide Befehle können auf das gr.clip Objekt angewendet werden.

gr.NewDL Array []

Ersetzt die anfängliche Anzeige-Liste mit einer Anzeigeliste die aus einer Anordnung von Objektnummern besteht. Null-Werte auf der neuen Anzeige-Liste werden als Null-Objekte behandelt. Null-Objekte werden weder gezeichnet werden, noch werden sie zu Laufzeitfehlern führen.
Sehen Sie sich das Unterthema Anzeigeliste in diesem Kapitel für weitere Erläuterungen an. Sehen Sie auch das Beispielprogramm, f24_newdl, für ein funktionierendes Beispiel dieses Befehls.

Audio Interface

Einführung

Das Audio Interface

BASIC! Android nutzt die Media Player-Schnittstelle zum Abspielen von Musik-Dateien. Diese Schnittstelle gehört nicht zu den stabilen Teilen des Android. Es gibt manchmal Unklarheit darüber, was es tut. Dies kann dazu führen, das zufällig erzwungene Abschalt- Ereignisse auftreten. Auch wenn diese Ereignisse selten sind, treten sie doch auf.

Audio Datei Typen

Der Musik-Player soll in der Lage sein, WAV, AAC, MP3, WMA, AMR, OGG-und MIDI-Dateien abzuspielen. Ich habe es mit MP3-und WAV-Dateien versucht. Die Leistung kann gegenüber anderen Dateitypen variieren.

Befehle

Audio-Dateien müssen in die Audio-Datei Tabelle geladen werden, bevor sie abgespielt werden können. Jede Audio-Datei in der
Audio Datei Tabelle hat einen eindeutigen Index, der durch den audio.load Befehl zurückgegeben wird.

audio.load <aft_nvar>, <filename_sexp>

Lädt eine Musik-Datei in die Audio-Datei Tabelle. Der Audio-Datei Tabellen Index wird in ‹aft_nvar› zurückgegeben. Die Datei muss im Verzeichnis ”/sdcard/rfo-basic/data/” oder einem seiner Unterverzeichnisse stehen. Sie können Dateien außerhalb des ”/sdcard/rfo-basic/data /” Pfades durch Angaben im Dateinamen erreichen. Zum Beispiel,
.. /.. / Musik / Blue Danube Waltz.mp3 würde den Zugang zu ”/sdcard / Musik / Blue Danube Waltz.mp3”schaffen.

audio.play <aft_nexp>

Wählt die Datei aus der Audio-Datei Tabelle, gekennzeichnet durch ‹aft_nexp› und beginnt sie zu spielen. Es darf keine Datei gespielt werden, wenn dieser Befehl ausgeführt wird. Wenn eine Datei gepielt wird, dann führen sie zuerst ein audio.stop aus. Das Abspielen der Musik stoppt, wenn das Programm nicht mehr läuft. Um einfach eine Musikdatei zu hören, ist es nötig das Abspielen und das Programm am Laufen zu halten. Eine Endlosschleife wird das erreichen:

```
Audio.load ptr, my_music.mp3
Audio.play ptr
```

```
Do
Pause 5000
Until 0
```

audio.stop

Stoppt die Wiedergabe der aktuell abgespielten Musikdatei. Der Befehl wird ignoriert, ist keine Datei abgespielt wird. Es ist am besten jedem audio.play Befehl einen audio.stop Befehl vorausgehen zu lassen.

audio.pause

Pause ist wie ein Stop, außer dass beim nächsten audio.play für diese Datei das Abspielen an dem Punkt fortgesetzt wird, wo der Wiederaufnahme
das Abspielen unterbrochen wurde.

audio.loop

Wenn in der aktuell wiedergegebenen Datei das Ende der Datei erreicht wird, dann wird wieder neu angefangen die Datei zu spielen.
Es muss eine aktuell wiedergegebene Datei geben, wenn dieser Befehl ausgeführt wird.

audio.volume <left_nexp>, <right_nexp>

Ändert die Lautstärke des linken und rechten Stereokanals. Es muss eine aktuell wiedergegebene Datei geben, wenn sein dieser Befehl ausgeführt wird.
Die Werte sollten zwischen 0,0 (niedrigste) bis 1,0 (höchste Lautstärke) reichen. Die Wahrnehmung des menschlichen Ohres verhält sich zum Niveau des dB-Wertes der Lautstärke- Änderungn entlang einer logarithmischen Skala. Das Ohr nimmt eine 10db Änderung als doppelt so laut wahr. Eine Änderung um 20db
würde jedoch 4 mal so laut erscheinen.
Ein 1-dB-Änderung würde etwa 0,89 sein. Eine Möglichkeit, einen Lautstärkeregler umzusetzen, wäre die Lautstärke so setzen, dass ein
Regler mit 1 dB Änderungen entsteht. Der folgende Code erstellt eine 16-Step-Tabelle (16 Stufen -Regler).

```
dim volume[16]
x =1
volume [1] = x
for i = 2 to 16
x = x * 0.89
volume [i] = x
```

next i

Ihr Code kann den Lautstärkewert aus der Tabelle für den Einsatz im audio.volume Befehl wählen. Am lautesten würde volume[1] sein.

audio.position.current <nvar>
Die aktuelle Position in Millisekunden der aktuell wiedergegebenen Hördatei wird in ‹nvar› zurückgegeben werden.

audio.position.seek <nexp>
Verschiebt die Wiedergabeposition der aktuell abgespielten Hördatei, an die durch ‹nexp› in Millisekunden angegebene Stelle.

audio.length <length_nvar>, <aft_nexp>
Gibt die Gesamtlänge der Datei in der Audio-Dateitabelle, die mit ‹aft_nexp› kennzeichnet ist zurück. Die Länge in Millisekunden wird in ‹length_nvar› zurückgegeben werden.

audio.release <aft_nexp>
Gibt die Ressourcen, die von der Datei in der Audio-Dateitabelle verwendet werden, mit ‹aft_nexp› zurück. Die Datei darf derzeit nicht wiedergegeben werden. Die angegebene Datei wird nicht mehr länger in der Lage sein, wiedergegeben zu werden.

audio.isdone <Boolean_nvar>
Wenn die aktuelle Datei noch wiedergegeben wird, dann wird ‹Boolean_nvar› auf Null gesetzt werden, anderenfalls wird es auf 1 eingestellt werden
Dies kann genutzt werden, um festzustellen, wann die Wiedergabe der nächsten Datei aus der Playliste starten soll.

```
Audio.play f[x]
Do
        Audio.isdone isdone
        Pause 1000
Until isdone
```

audio.record.start <fn_svar>
Starten Sie eine Audio-Aufnahme über das Mikrofon als Audioquelle. Die Aufnahmen werden im Gerät gespeichert in der angegebenen Datei. Die Datei muss die Erweiterung .3GP haben. Die Aufnahme wird fortgesetzt, bis der audio.record.stop Befehl

ausgegeben wird.

audio.record.stop
Stoppt die zuvor gestartete Audio-Aufnahme.

SoundPool

Einführung
Ein SoundPool ist eine Sammlung von kurzen O-Tönen, die vorgeladen und bereit für sofortige Spiel sind. SoundPool O-Töne können gleichzeitig und übereinander gespielt werden. Sie können sie auch spielen über Melodie einer aktuell abgespielten Sound-Datei, die mit meines Hilfe Audio.play Programms gespielt wird.
In einem Spiel, kann die Audio.play Datei die Hintergrundmusik sein, während die O-Töne des SoundPool die Game-Sounds sein würden (Bang, Pow, Screech, etc).
Ein SoundPool wird geöffnet mit dem Befehl SoundPool.open. Nachdem der Soundpool geöffnet wurde, werden die Sound Stücke in den Speicher von Dateien mit dem Befehl SoundPool.load geladen werden. Geladen O-Töne können nun gespielt und immer wieder wiederholt werden, mit dem Befehl SoundPool.play.
Ein Abspielen von Sound nennt man einen Sound-Stream. Individuelle Sound-Streams können angehalten (SoundPool.pause) werden, einzeln und als eine Gruppe, wieder fortgesetzt (SoundPool.resume) und gestoppt (SoundPool.stop). Andere Stream Parameter (Priorität, Volumen und Geschwindigkeit) können im laufenden Betrieb geändert werden.
Die SoundPool.release Befehl schließt den SoundPool. Ein neuer SoundPool kann dann geöffnet werden in einer anderen Phase des Spiels. SoundPool.release wird automatisch aufgerufen, wenn der Programmablauf beendet ist..

Befehle

Soundpool.open <MaxStreams_nexp>
Die MaxStreams Ausdruck gibt die Anzahl der Soundpool-Streams an, die auf einmal gespielt werden können. Wenn die Anzahl der abgespielten Streams diesen Wert überschreitet, wird die niedrigste Priorität des Streams beendet werden.
Hinweis: Ein Stream der über audio.play gespielt wird, wird nicht als Soundpool Stream gezählt.

Soundpool.load <soundID_nvar>, <file_path_sexp>
Die angegebene Datei wird geladen. Ihre Klang-ID wird in der angegebenen Variablen zurückgegeben. Die Sound-ID wird verwendet, um den Klang zu spielen und auch zum

Entladen des Tons. Der Sound-ID wird als Null zurückgegeben, wenn die Datei aus irgendeinem Grund nicht geladen wurde.
Die Standard-Datei-Pfad ist ”sdcard /rfo-basic/data/”
Hinweis: Es kann einige hundert Millisekunden dauern bis der Sound geladen wurde. Legen Sie eine ”Pause 500”-Anweisung nach dem load ein, wenn Sie den Sound sofort spielen möchten.

Soundpool.unload <soundID_nexp>
Der angegebene geladene Sound wird entladen.

Soundpool.play streamID (nvar), soundID, right_volume, left_volume, Priorität, Schleife, Rate
Startet den angegebenen Sound mit der ID zum abspielen.
Die Sound-ID steht in der numerischen Variablen - streamID. Wenn der Stream nicht gestartet wurde, wird der Wert der zurückgegeben wird Null sein. Die Stream-ID wird auch verwendet, um nach einer Pause, weiter zu spielen und beenden auch den Stream. Sie wird auch verwendet in den Stream- Modifikations Befehlen (Soundpool.setrate, Soundpool.setvolume, Soundpool.setpriority und Soundpool.setloop).
Die linken und rechten Lautstärke-Werte müssen im Bereich von 0 bis 0,99 mit Null sein um still gestellt zu sein.
Die Priorität ist ein positiver Wert größer oder gleich Null. Die niedrigste Priorität ist Null.
Der Schleifen-Wert von -1 spielt eine Schleife mit dem Ton für immer. Andere Werte als -1, geben an wie oft der Stream abgespielt wird. Ein Wert von 1 wird den Stream zweimal spielen. Der Rate-Wert ändert die Wiedergabe des Sounds ab. Der normale Satz beträgt 1. Der Mindestsatz (langsam) ist 0.5. Die maximale Geschwindigkeit (schnell) ist 1.85.

Soundpool.setvolume <streamID_nexp>, <leftVolume_nexp>, <rightVolume_nexp>
Ändert die Lautstärke eines laufenden Streams.
Die linken und rechten Lautstärke-Werte müssen im Bereich von 0 bis 0,99 mit Null sein, um still gestellt zu sein.

Soundpool.setrate <streamID_nexp>, <rate_nexp>
Ändern Sie den Play-Rate von einem laufenden Stream.
Der normale Satz beträgt 1. Der Mindestsatz (langsam) ist 0,5. Die maximale Geschwindigkeit (schnell) ist 1.85.

Soundpool.setpriority <streamID_nexp>, <priority_nexp>
Ändert die Priorität eines laufenden Stream.

Die niedrigste Priorität ist Null.

Soundpool.pause <streamID_nexp>
Unterbricht das Abspielen des angegebenen Stream. Wenn die Stream ID Null ist, werden alle Datenströme angehalten.

Soundpool.resume <streamID_nexp>
Setzt das Abspielen des angegebenen Streams fort. Wenn die Stream-ID Null ist, werden alle Ströme wieder aufgenommen werden.

Soundpool.stop <streamID_nexp>
Stoppt die Wiedergabe des angegebenen Stream.

Soundpool.release
Schließt das SoundPool und gibt alle Ressourcen frei. Wird Soundpool.open aufgerufen, kann ein neuer SoundPool geöffnet werden.

GPS
Diese Befehle ermöglichen den Zugriff auf die Roh-Standortdaten von einem Android-Gerät, welche die GPS Hardware zur Verfügung stellt
Bevor Sie versuchen, diese Befehle verwenden, stellen Sie sicher, dass GPS im Android - Setting Programm eingeschaltet ist.
Die Beispiel- Programmdatei f15_gps.bas zeigt ein Beispiel für die Verwendung der GPS-Befehle.

Befehle

gps.open
Schaltet die GPS-Hardware ein und startet die Berichterstattung zu Standortinformationen. Dieser Befehl muss vor der Anwendung anderer GPS-Befehle verwendet werden.

gps.close
Schaltet das GPS-Hardware aus und stoppt die Standortbestimmung. GPS wird automatisch geschlossen, wenn Ihr BASIC! Programm aufhört. GPS ist nicht aus, wenn Sie die HOME-Taste drücken, während das GPS-Programm läuft.

gps.provider <svar>
Gibt den Namen des GPS-Anbieter in ‹svar› an.

gps.accuracy <nvar>
Returns the accuracy level in ‹nvar›.

Gibt die Genauigkeit in ‹nvar› an.

gps.latitude <nvar>
Gibt die geografische Breite in Dezimalgrad in ‹nvar›an.

gps.longitude <nvar>
Gibt die geographische Länge in Dezimalgrad in ‹nvar› an.

gps.altitude <nvar>
Gibt die Höhe über NN in Meter in ‹nvar› an.

gps.bearing <nvar>
Gibt das Lager in ‹nvar›an.xxxx

gps.speed <nvar>
Gibt die Geschwindigkeit in Meter pro Sekunde in ‹nvar›an.

gps.time <nvar>
Gibt das gibt die UTC-Zeit in Millisekunden seit dem 1. Januar 1970 an.

Sensoren

Einführung
Android-Geräte können mehrere Arten von Sensoren haben. Die für Android vordefinierten Sensoren sind:
Accelerometer,(Beschleunigung) Type = 1
Magnetfeld Type = 2
Orientierung, Type = 3
Gyroskop, Type = 4
Light, Type = 5
Luftdruck, Type = 6
Temperatur, Typ = 7
Proximity, Type=8
Gravitations- Schwerpunkt, Type = 9
Lineare Beschleunigung Type = 10
Rotationsvektor Type = 11

Einige Details über die (meisten) dieser Sensoren kann auf der Webseite Android-Sensor Event gefunden werden
(http://developer.android.com/reference/android/hardware/SensorEvent.html) .

Nicht alle Android-Geräte haben alle diese Sensoren. Einige Android-Geräte besitzen keinen dieser Sensoren. Die BASIC! Befehl, sensors.list, kann verwendet werden, um eine Bestandsaufnahme der Sensoren auf Ihrem Gerät zur Verfügung zu stellen.
Einige neuere Geräte können Sensoren haben, die derzeit nicht von Basic! unterstützt werden. Diese Sensoren werden im Bericht mit ”Unknown (Unbekannt), Type = NN” angezeigt, wobei NN die Sensor-Typ-Nummer ist.

Sensor-Befehle

sensors.list list$[]

Dieser Befehl bietet eine Liste der verfügbaren Sensoren auf einem bestimmten Android-Gerät. Der Parameter, Liste $ [], muss ein un-dimensioniertes Array sein. Die Liste wird in dieses Array zurückgegeben. Die Elemente enthalten die Namen und Typen der verfügbaren Sensoren. Zum Beispiel, ”Gyroscope, Type = 4”. Der folgende Programmausschnitt kann verwendet werden, um die Elemente der Liste $ [] aufzuführen.

```
sensors.list list$[]
array.length size, list$[]
for index = 1 to size
        print list$[ index ]
next index
end
```

Der sensors.open Befehl sollte nicht vor dem Ausführen dieser Befehle ausgeführt werden.

sensors.open {t1 t2, , tn}

Öffnet eine Liste mit Sensoren zum Lesen. Die Parameterliste besteht aus den Typ-Nummern der Sensoren die geöffnet werden sollen.
Zum Beispiel, ”sensors.open 1, 3”, würde den Beschleunigungs Typ 1 und den Orientierungs. Typ 3 Sensor öffnen. Dieser Befehl muss vor der Erteilung des sensors.read Kommandos ausgeführt werden. Sie sollten nur dann die Sensoren öffnen, die Sie tatsächlich lesen wollen. Jeder Sensor belastet die Batterie und erhöht die Hintergrund-CPU-Auslastung.

sensors.read sensor_type, p1, p2, p3

Dieser Befehl gibt die aktuellen Werte aus den Sensoren an, die von den ”sensor_type” Parameter ausgewählt wurden. Der Typ-Parameter muss im Bereich von 0 bis 9 sein. Die Werte werden wieder in die P1, P2, und P3 Parameter platziert. Die Bedeutung dieser

Parameter ist abhängig davon wie der Sensor ausgelesen wird. Nicht alle Sensoren werden drei Parameter zurückgeben. In jenen Fällen, werden die nicht genutzten Parameter-Werte auf Null gesetzt. Sehen Sie die Android-Sensor Event-Webseite zur Bedeutung dieser Parameter.

sensors.close
Schließt die zuvor geöffnete Sensoren. Durch Sensor- Hardware Abstellen kann der Batterieverbrauch verhindert werden. Sensoren werden automatisch geschlossen, wenn der Programmablauf über die BACK-Taste oder Menü-› Stop gestoppt wird.

Superuser
Diese Kommandos bieten für die Ausführung von Befehlen auf Superuser gerootete Geräte. Finden Sie im Beispielcode-Programm, f36_superuser.bas, ein Beispiel für die Verwendung dieser Befehle.

Befehle

Su.open
Anfragen Superuser-Berechtigung.

Su.write <sexp>
Führt einen Superuser-Befehl aus.

Su.read.ready <nvar>
Tests für Antworten von einem Su.write Befehl. Wenn das Ergebnis nicht Null ist, dann sind Antwortleitungen zur Verfügung.
Nicht alle Befehle als Superuser geben eine Antwort zurück. Wenn keine Antwort nach ein paar Sekunden zurückgegeben wird, nachdem es sein sollte, kann angenommen werden, dass es keine Reaktion gab.

Su.read.line <svar>
Platziert die nächste verfügbare Antwortzeile in die String-Variable.

Su.close
Beendet den Superuser-Modus.

Anhang A - Liste der Befehle

(die Seitenangaben stimmen nur annähernd mit den Seiten dieser Übersetzung überein)

Anhang B - Beispielprogramme

Die Beispielprogramme werden in ”/sdcard/rfo-basic/source/Sample Program” neu geladen, wenn eine neue Version von BASIC! installiert wird. Sie können darauf zugreifen, indem Sie auswählen Menu › Load. Clicken Sie auf die ”Sample Programs” Zeile. Die Beispielprogramme werden aufgelistet und können geladen werden. Wenn Sie eines dieser Programme laden und speichern, sollte das Programm im ”/sdcard/rfo-basic/source/” und nicht in ”/sdcard/rfo-basic/source/ Sample Program” gespeichert werden.
Sie können BASIC! dazu zwingen diese Beispielprogramme neu zu laden:
Verwenden Sie dazu den BASIC! Delete(Löschen) Menübefehl, wechseln Sie in das Verzeichnis ”/sdcard/rfo-basic/source/ Sample Program” und löschen Sie die Datei”f01_vxx.xx_read_me file”
Beenden Sie BASIC! über das Menü -› More -› Re-enter BASIC!

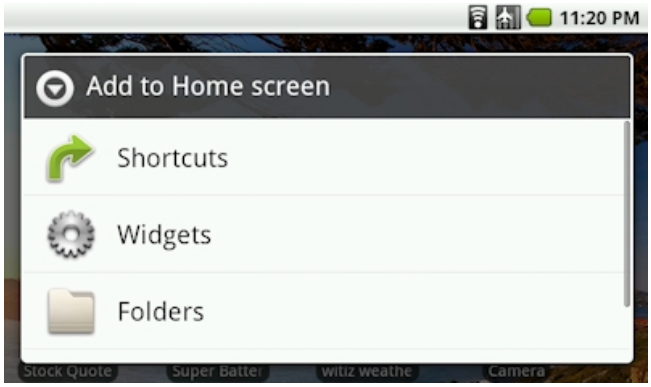
Anhang C - Launcher Short Cuts Tutorial

Einführung

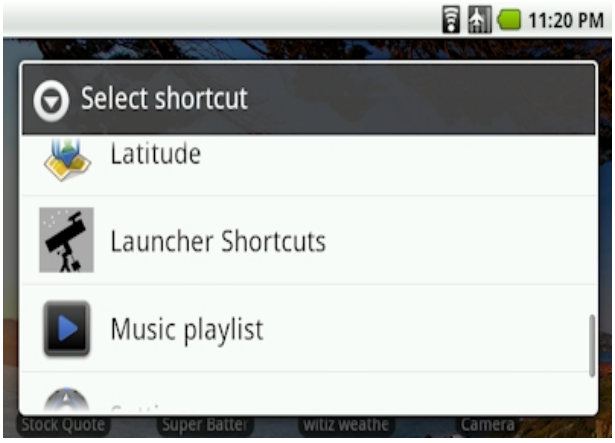
Dieses Tutorial will erklären wie Sie aus einem BASIC! Programm eine ”Anwendung” für die Android- Benutzeroberfläche ihres Gerätes erstellen.
Diese ”Anwendung” bekommt ein eigenes Symbol und eigenen Namen. Der offizielle Name für ein Android Icon und diese Art von ”Anwendungen” ist ”Shortcut”. Die BASIC! Basissoftware muss jedoch installiert sein, um damit arbeiten zu können.
Es gibt auch eine Option, mit der tatsächlich eine eigenständige Anwendung als .apk-Datei gebaut werden kann, bei der es nicht erforderlich ist, zusätzlich die BASIC! Basissoftware zu installieren. Der Prozess ist schwieriger, es wird aber dadurch eine Anwendung entstehen, die in der Folge im Android Market angeboten werden kann. Siehe Anhang D.

Das Verfahren zur Herstellung einer Anwendungsverknüpfung :

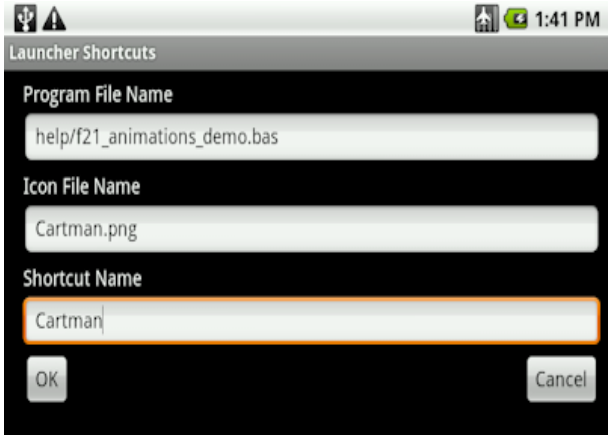
1. Starten Sie BASIC!
2. Wählen Sie unter Menü-› More-› EXIT zum Verlassen BASIC!
3. Längerer Druck auf den Android- HOME- Bildschirm
4. Sie sollten etwa das folgende Bild sehen.



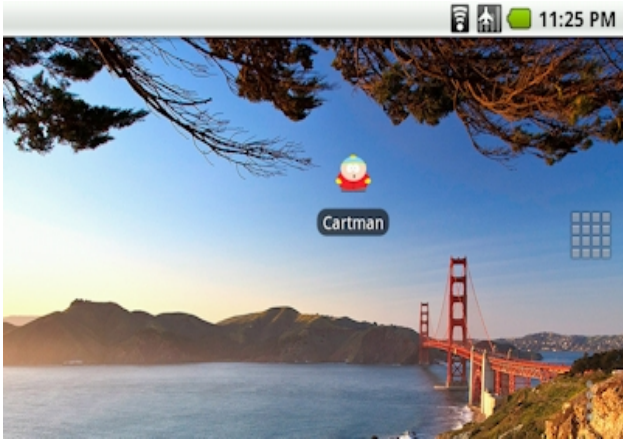
5. Tippen Sie auf Shortcuts
6. Blättern Sie in der Select Shortcut Seite, bis Sie das BASIC! Symbol mit dem Launcher Shortcuts Label sehen.



7. Tippen Sie auf das BASIC! Icon.
8. Dieser Bildschirm erscheint.



9. Füllen Sie das Formular genau so aus , wie dargestellt.
10. Drücken Sie auf OK
11. Sie sollten so etwas wie das, auf Ihrem Startbildschirm sehen



12. Tippen Sie auf das Cartman Shortcut
13. BASIC! wird beginnen und führt das Cartman Jumping Demo aus.

Was Sie wissen müssen:

1. Die (Shortcut) Symbol Image-Datei muss sich im Verzeichnis ”/sdcard /rfo-basic /data/” befinden.
2. Das Programm, dass Sie als ”Anwendung” laufen lassen wollen, muß im ”source”-Verzeichnis oder einem seiner Unterverzeichnisse stehen. In diesem Beispiel befindet sich die Programmdatei in dem Unterverzeichnis samples(d) vom Verzeichnis ”source(d)”.
3. Das Symbol sollte eine . png-Datei sein. Eine Google-Suche für ”Icon” wird tausende freier Symboledatein anbieten. Kopieren Sie das gewünschte Symbol in das Verzeichnis ”rfo-basic/data” auf der SD-Karte.
4. Seien Sie sehr vorsichtig, bei der korrekten Schreibweise der Namen für die Programm-und Icon-Dateien. BASIC! wird nicht überprüfen, ob diese Dateien tatsächlich während des ”compile”-Prozess existieren. Wenn Sie den Namen einer Symbol-Datei angeben, die nicht existiert, wird Ihre Verknüpfung nur das generische Android-Symbol haben. Wenn die Programmdatei mit dem von Ihnen angegebenen Namen nicht existiert, wenn Sie das Verknüpfungs- Icon antippen, erhalten Sie eine Fehlermeldung in Form einer Programm- Datei im Editor zu sehen.
5. Die Shortcut-Name sollte neun (9) Zeichen lang sein. Android zeigt nicht mehr als neun Zeichen an.
6. Sie können beliebig viele Verknüpfungen, soviel Ihr(e) Home-Bildschirm(e) aufnehmen

können, erstellen.
7. Durch Drücken auf ”Abbrechen” im Launcher Shortcuts Dialog können Sie einfach den Vorgang abbrechen und zurück zu dem Home-Bildschirm gelangen.
8. Wenn Sie ein Programm mit einem Launcher Shortcut im BASIC! Editor verwenden, sollten Sie dieses jedoch immer über das BASIC! Menü-› Mehr-› Exit - beenden.
Wenn ein über das Launcher Shortcut gestartetes Programm läuft, drücken noch einmal oder zweimal die BACK- Taste zum Verlassen und BASIC! wird zum Home-Bildschirm zurückkehren.

Anhang D - Der Aufbau einer Standalone-Anwendung

Hinweis:

Ein BASIC! Benutzer, Nicolas Mougin, hat ein automatisches Tools zur Erstellung einer Standalone-Anwendung (APK) entwickelt. Dieses Werkzeug kann heruntergeladen werden unter.
http://mougino.free.fr/rfo-basic-app-builder.zip
Mit Herrn Mougin's Tool vermeiden Sie die im Folgenden beschriebenen Arbeiten.
Haben Sie Fragen oder Probleme zu diesem Tool können Sie Herr Mougin und andere Nutzer des Werkzeugs im BASIC!- Forum unter diesem Thread kontaktieren:
http://rfobasic.freeforums.org/rfo-basic-app-builder-f20.html

Einführung

Dieses Dokument zeigt, wie eine Standalone-Anwendung aus einem BASIC! Programm erstellt wird. Die entstehende Anwendung benötigt kein zusätzlich installiertes BASIC! um zu laufen. Sie bekommt einen eigenen Namen und ein Programmsymbol und kann so im Android Market oder anderswo angeboten werden.
Das Verfahren beinhaltet die Einrichtung der Android-Entwicklungsumgebung und ein paar einfache, gezielte Änderungen an dem BASIC! Java-Quellcode.

Lizenz-Information

BASIC! wird unter den Bedingungen der GNU General Public License vertrieben.
Eine Auswirkung davon ist, dass der Quellcode für Ihre Anwendung für jeden, der Sie fragt, einsehbar sein muss.

Einrichten der Entwicklungsumgebung

1. Downloaden und installieren Sie die neueste Version des Java Development Kit (JDK). Suchen Sie über Google "java jdk download" und gehen Sie auf die Download-Site von oracle.com. Nicht von einer anderen Website herunterladen. Hinweis: Der Download JDK enthält die Java ausführende Time Environment (JRE) die auch benötigt wird.
2. Das Entwicklungs - Android SDK-Installationsprogramm herunterladen, von: http://developer.android.com/sdk/index.html
3. Weiter mit: http://developer.android.com/sdk/installing/index.html

Führen Sie das SDK- Installationsprogramm aus. Installieren Sie die empfohlenen Produkte.
(Windows: Wenn Sie die Nachrichten erhalten, dass nichts installiert wurde: den SDK Manager zu schließen. Gehen Sie zu Start › Alle Programme -› Android SDK-Tools. Klicken Sie rechts auf den SDK Manager und wählen Sie ausführen als Administrator).
Wenn Sie eine Anforderung zum Starten der ADB erhalten, tun Sie es.
Schließen Sie den SDK Manager, wenn alle Pakete installiert wurden.

(4) Herunterladen und Installieren der Eclipse Integrated Development Entwicklungsumgebung (IDE) von: http://laughton.com/basic/eclipse. Wählen Sie die 32-Bit oder 64-Bit-Version von je nach Ihrem Entwicklungscomputer. Andere Versionen von Eclipse können Sie auch verwenden, aber diese Anweisungen funktionieren möglicherweise nicht und somit sind Sie auf Sie selbst gestellt.

5. Weiter auf: http://developer.android.com/sdk/installing/installing-adt.html
Starten Sie Eclipse. Folgen Sie die Anweisungen zur Installation der ADT.
Akzeptieren Sie alle Sicherheitswarnungen und ermöglichen Sie Eclipse neu zu starten, nach den Anweisungen. Fahren Sie mit ”Configure the ADT.” fort. Download der neuesten API-Ebene, egal welche API das Niveau Ihres Android-Geräts ist.
Ignorieren Sie ”Updating the ADT” und den Rest der Seite.
Ändern Sie die API-Ebene, die Überprüfung von Fehlerwarnungen:
Wählen Sie Window-›Preferences-›Android-›Lint Error Checking
Finden Sie ”NewApI” zu, und klicken Sie darauf.
Ändern Sie in der Dropdown-Liste den Schweregrad der Fehler Warnung.
Drücke Apply (Anwenden).

Download des Standalone- Anwendungs Source Code

Gehe zu: http://laughton.com/basic/versions/index.html und klicken Sie auf die höchste Version Nummer. Suchen Sie nach der Überschrift ”Source Code For Building A Stand Alone Application”. Klicken Sie auf ”here” zum download der neuesten ”MyApp.zip” Entpacken Sie diese Datei in einen Ordner namens TestMyApp. Er enthält dann eine readme.txt-Datei und ein Verzeichnis MyApp. MyApp enthält den Quellcode in Form eines Eclipse-Projekts. Die Inhalt der Dateien in diesem Ordner wird geändert werden, wenn Sie den Code mit Eclipse ändern. Sie sollten immer einen anderen Ordner für jede neue Anwendung verwenden, die Sie aus MyApp erstellen. Die readme.txt-Datei enthält

alle Informationen, die eindeutig im Zusammenhang mit dieser bestimmten Version stehen. Insbesondere wird es die Quellcode-Dateien auflisten, die seit der vorherigen Version geändert wurden. Diese Informationen sind nützlich beim Aktualisieren der Anwendung für die Verwendung einer neuen BASIC-Version! Sie werden damit eine Reihe von Änderungen an dem ursprünglichen MyApp Source-Code machen können. Die neuen Versionen von BASIC! haben aber meist nur eine Quelldatei, die sich geändert hat. Es gibt keine Notwendigkeit, alle ursprünglichen MyApp Änderungen zu wiederholen, wenn nur eine Datei in einer neuen Version geändert wurde. Sie können nur dazu nur eine Datei von der neuen Version Quelle kopieren und Sie ihn in Ihre MyApp einfügen. Eine Folge dieses Prozesses ist, dass Sie wahrscheinlich in einem anderen Verzeichnis den MyApp Originalordner erhalten, den Sie als Vorlage für zukünftige MyApp-Ordner benötigen.

Erstellen eines neuen Projekts in Eclipse

Select File -› New -› Project . . . -› Android -› Android Project vom existierenden Code.
Im Dialogfeld Import Projekte navigieren Sie zu dem TestMyApp Ordner(directory) der den entpackten Ordner MyApp enthält.
Überprüfen Sie das Projekt: com.rfo.myapp.Basic. Prüfen Sie nicht alle anderen Dialogfelder. Klicken Sie auf Fertig stellen.

Eine APK aus MyApp generieren

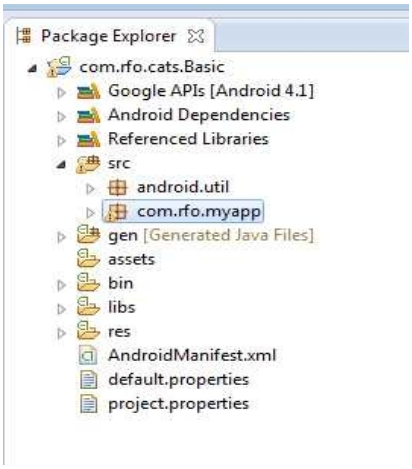
Stellen Sie zunächst sicher, dass alle Dateien, die Sie geändert gespeichert wurden. Die APK-Erstellungsprozess speichert nicht automatisch die geänderten Dateien.
Auf der linken Seite des Bildschirms mit der rechten Maustaste auf ”com.rfo.myapp.Basic” clicken.
Wählen Sie: Android Tools -› Export Signed Application Package (Export signiertes Anwendungspaket)
Klicken Sie auf Next (weiter) im Dialogfeld Projekt Checks.
Wählen Sie ”Create New Keystore.” (Erstelle neuen Schlüsselspeicher)
Geben Sie Passwort ein.
Klicken Sie auf Next (weiter.
Füllen Sie das Dialogfenster Key Creation (Schlüssel erzeugen) aus.
Wählen Sie einen beliebigen Namen für ein Alias.
Geben Sie 23 Jahre Validity (Gültigkeit) ein

Klicken Sie auf Next (weiter)
Im Dialogfeld Destination(Ziel) und Key/Zertifikat überprüfen,
Suchen Sie den Ordner, in den Sie die APK setzen möchten.
Name der APK ”TestMyapp”
Klicken Sie auf Fertig stellen (Finish)
Jetzt installieren und Ausführen von TestMyApp.apk
Hinweis zu diesem Schlüsselspeicher: Sie verwenden diese gleiche Schlüsselspeicher für alle Apks die Sie erstellen. Merken Sie sich, welchen Namen Sie verwendet haben und wo der Ort der Datei ist und was die Kennwörter sind.
Wir können nun die Anpassung von MyApp verschieben, wenn alles hat gut bis zu diesem Punkt gegangen ist.

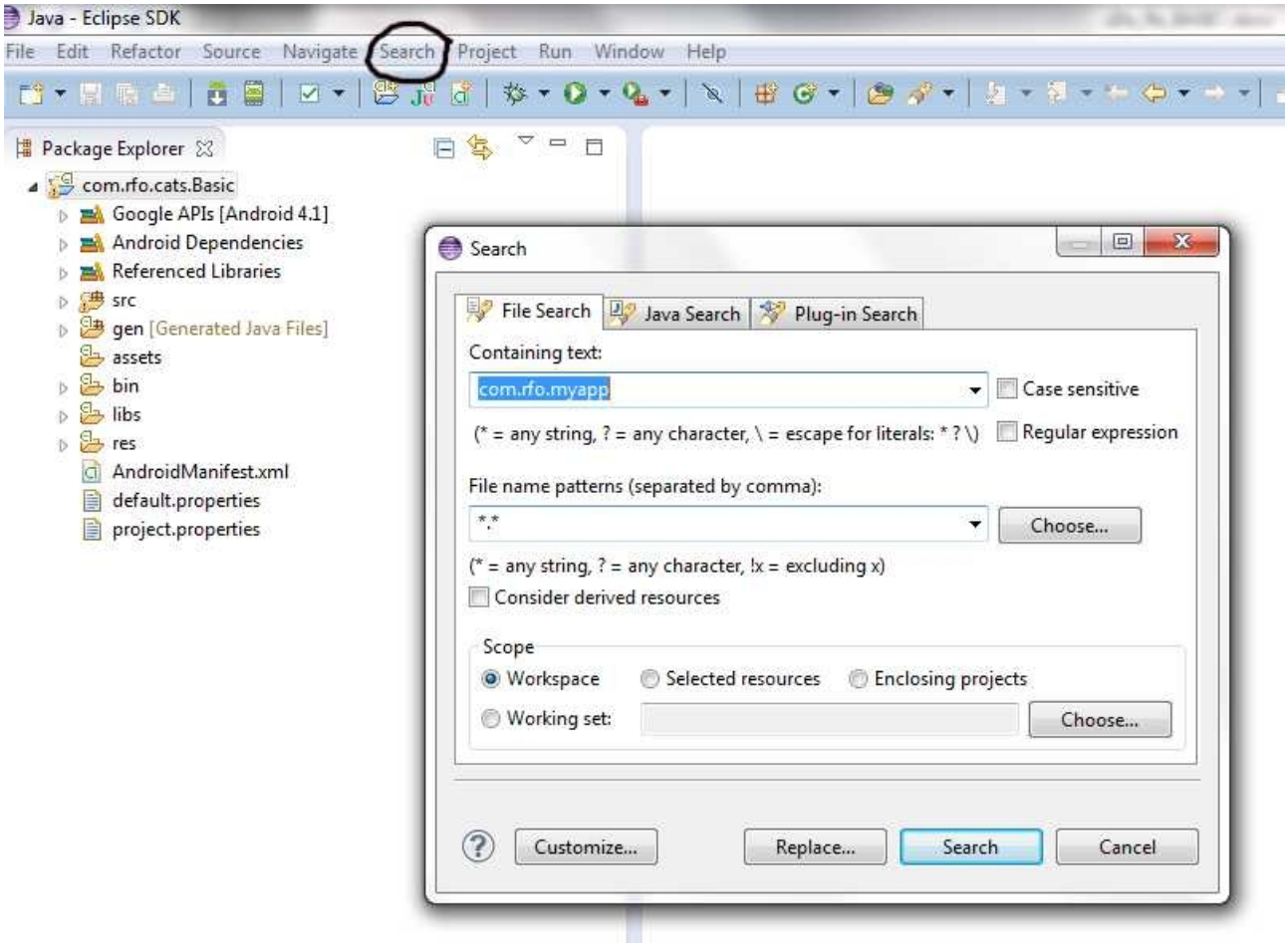
Benennen Sie das Paket um

Der Paketname ist das, was Ihre Anwendung von jeder anderen unterscheidet.
Egal, wie Sie Ihre Anwendung benennen der Paketname wird von Android verwendet, um Ihre Anwendung zu identifizieren. Im Paket- Explorer auf der linken Seite der Seite antippen: ”rfo.com.myapp.BASIC”.

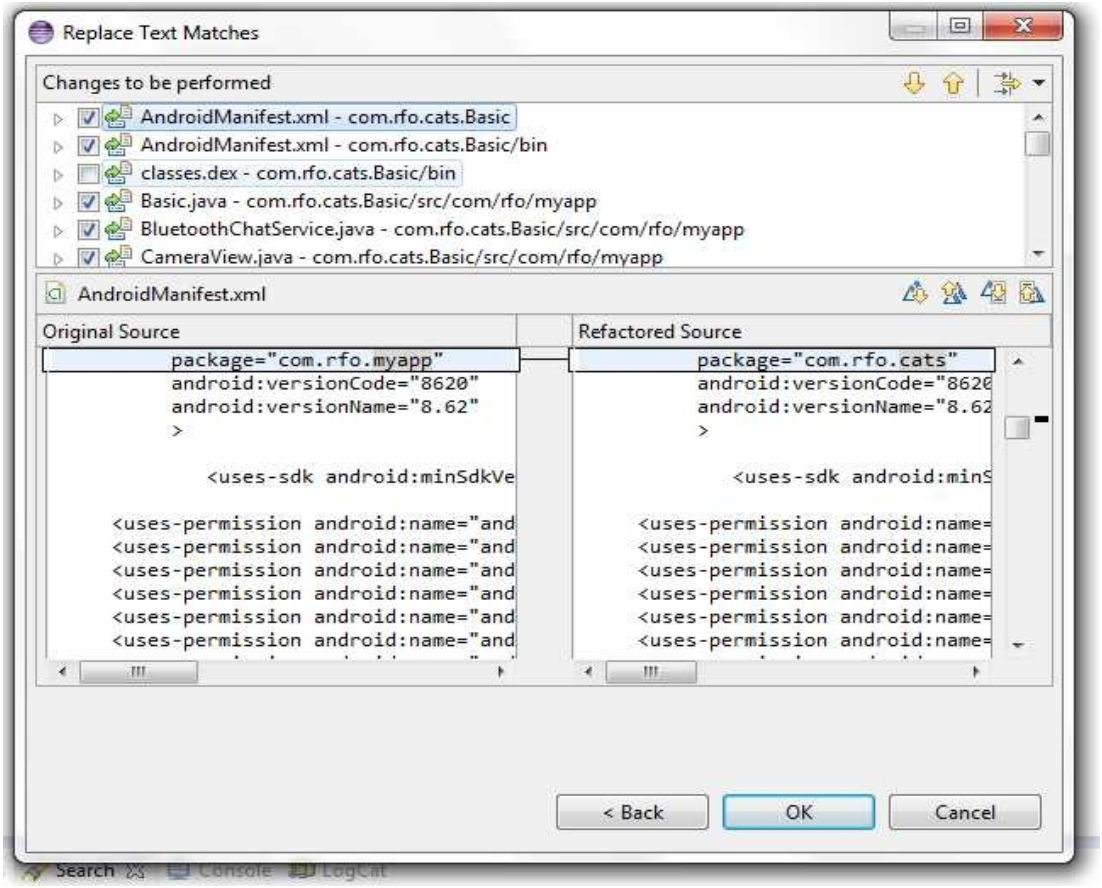
Nächste Auswahl: File (Datei) › Rename (umbenennen)
Geben Sie den neuen Namen für das Paket ein. Angenommen, Ihre Anwendung soll ”cats” genannt werden, dann ändern Sie den Namen des Pakets zu ”rfo.com.cats.BASIC”
Stellen Sie sicher, dass das Kontrollkästchen Update References (Verweise aktualisieren) aktiviert ist.
Drücke OK.
Im Package Explorer auf der linken Seite der Seite, klicken Sie auf Open zum Öffnen der ”src”- Hierarchy, wie abgebildet.



Wählen Sie ”com.rfo.myapp”, dass es markiert ist.
Wählen Sie File-›Rename und benennen Sie es um zu ”com.rfo.cats”. Stellen Sie sicher, dass das Kontrollkästchen Update References (Verweise aktualisieren) aktiviert ist.
Vom der Menu Bar, wählen Sie: Search-›File.
Füllen Sie den Dialog wie folgt:



Drücke Replace (Ersetzen)
Das Dialogfeld Ersetzen Text Matches wird angezeigt.
Geben Sie ”com.rfo.cats” in das ”With:” Feld ein.
Drücke -Preview (Vorschau).



Deaktivieren Sie das Kontrollkästchen für ”classes.dex”, wie gezeigt.
Drücken Sie OK
Ein Launch Configuration Change (Änderungs)-Dialogfeld wird angezeigt. Drücken Sie Yes (ja).

Schließlich wählen Sie: Project -› Clean
Drücken Sie auf OK im Dialogfeld Clean.
Das ist der Zeitpunkt an dem das Paket erfolgreich umbenannt wurde. Als nächstes machen wir die Änderungen die nötig sind, um Ihr BASIC! Programm auszuführen.

Installation eines BASIC! Programms in der Anwendung

Außerhalb von Eclipse

Öffnen Sie das BASIC! Programm in einem Texteditor

Wenn Ihr Programm INCLUDED- Dateien nutzt, sollten Sie mit Text-Merge (Text anfügen) die enthaltenen (included) Dateien in einer einzigen Text-Datei zusammenfassen.

Kopieren Sie das gesamte Programm in die Zwischenablage

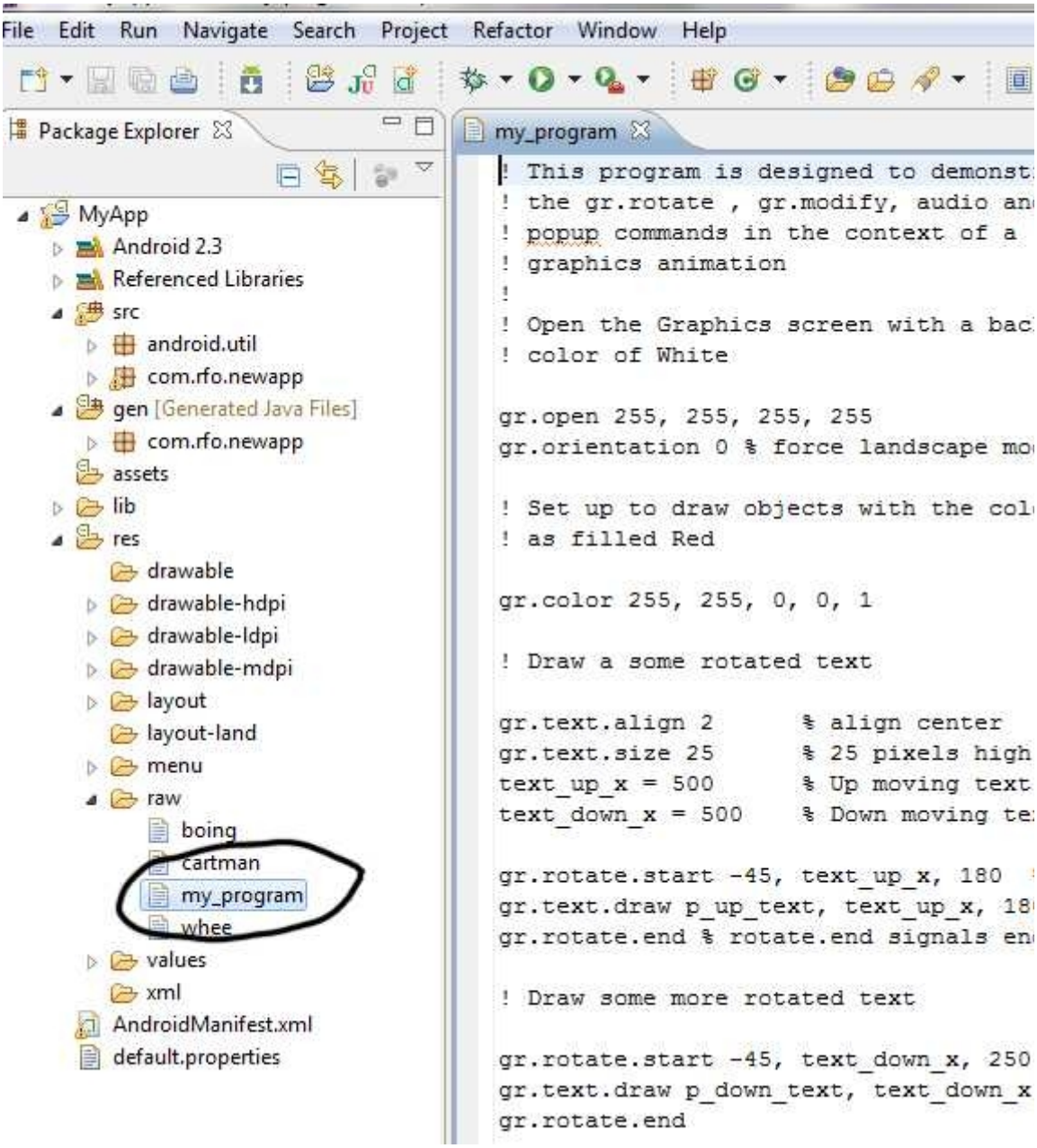
Im Package Explorer auf der linken Seite,

Öffnen Sie - res

Öffnen Sie - raw

Klicken Sie doppelt auf my_programm

Die Datei wird geöffnet



Drücken Sie auf Edit - › Select All (Bearbeiten -› Alles auswählen)
Drücken Sie auf Edit -› Paste (Bearbeiten -› Einfügen)
Drücken Sie die Taste ”X” auf der Registerkarte von my_programm, um die Datei zu schließen.
Drücken Sie ”Ja” auf die Speichern-Dialogbox Ressource.

Ausführen der Änderungen an Basic.Java

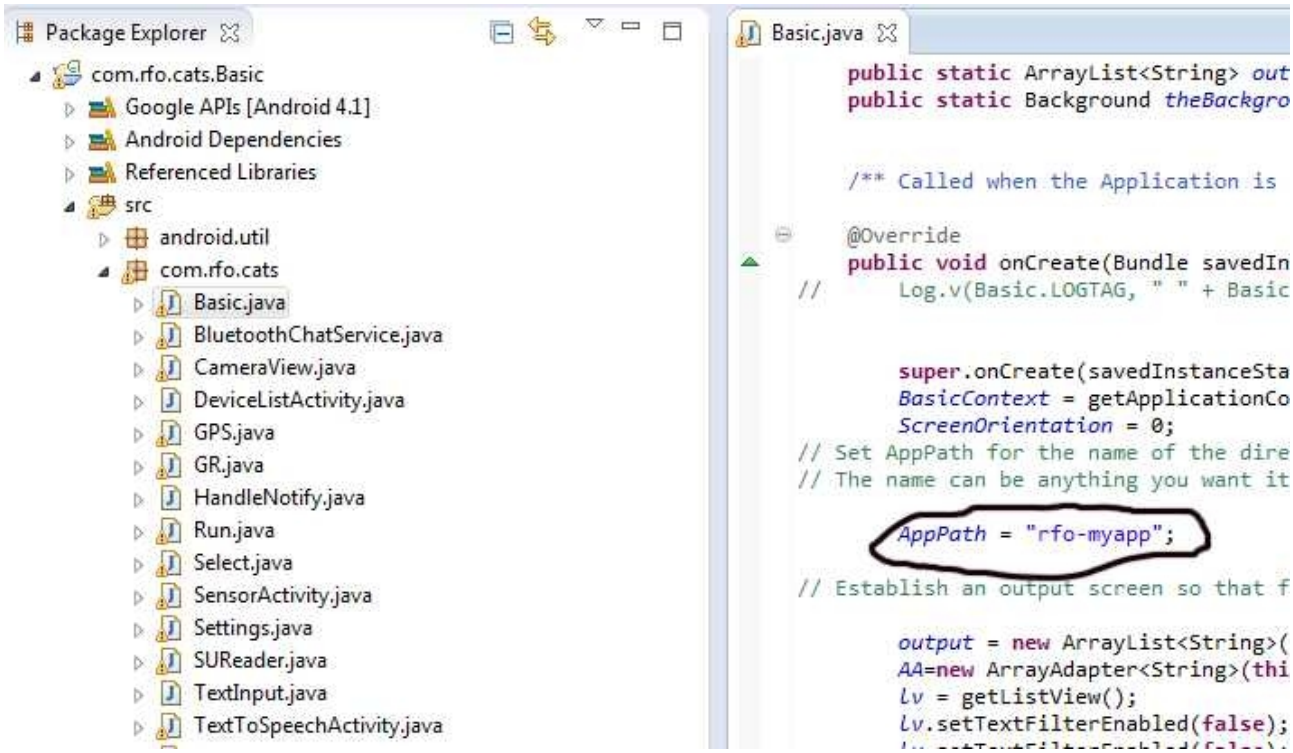
Basic.java ist die Java-Klasse, welche die Ausführung von BASIC! startet.
Sie empfängt control (Kontrolle) aus dem Android Launcher.
Die Funktion von Java ist, die Dinge zu initialisieren.
Zu diesen Dingen gehören das Einrichten des Anwendung Verzeichnisses auf der SD-Karte, das Laden Ihres Basic-Programm in den Ausführungs- Speicher und das Laden beliebiger Bild-oder Audio-Dateien, die notwendig sein können.
Sobald das erledigt ist, wird control (die Steuerung) übertragen an Run.java (das Starten von Java), um den eigentlichen Ablauf des Programms auszuführen.

Öffnen Sie die Basic. java-Datei

Im Package Explorer auf der linken Seite, öffnen Sie com.rfo.newapp.
Doppelklicken Sie auf Basic.java um die Datei zu öffnen.

Benennen Sie die Anwendung im Top-Level-Datei-Verzeichnis

Wenn Ihre Anwendung eine Datei I/O (Eingabe/Ausgabe) benutzt, wird sie ein anwendungsspezifisches Verzeichnis im Root-Verzeichnis der SDCARD brauchen.



Der Name dieses Top-Level-Verzeichnises wird von dem Wert in der AppPath Variable gesetzt. Ändere ”rfo-myapp” zu ”rfo-cats”
Zwei Unterverzeichnisse werden in dem Verzeichnis erstellt werden.
Das ”data” Daten-Unterverzeichnis wird für die meisten Ihrer Dateien verwendet.
Das ”databases” Datenbanken-Verzeichnis wird für etwaige SQLITE-Datenbanken verwendet werden. Diese Verzeichnisse werden erstellt und initialisiert von der InitDirs()-Methode wie Sie oben gesehen haben.

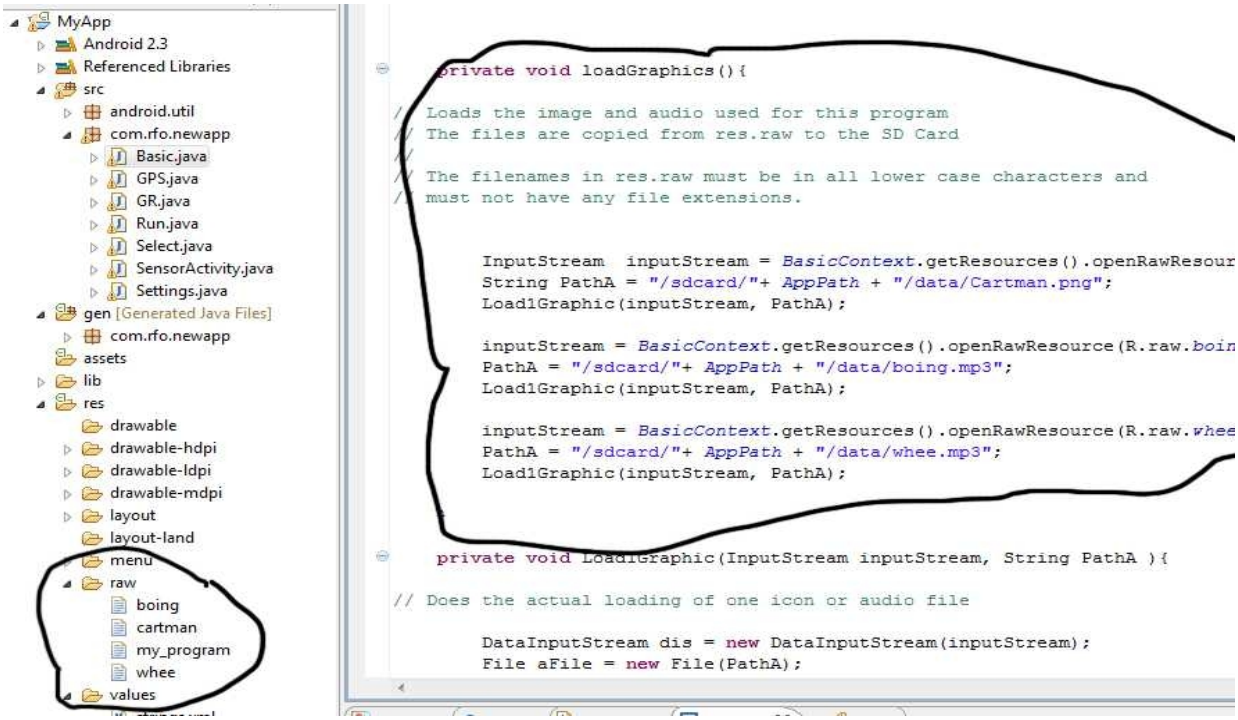
Erstellen von Bild-und Audiodateien

Es gibt zwei Möglichkeiten, den Zugriff auf Bilder, Audio und andere Dateien in Ihrem APK zu realisieren. Sie können entweder die Dateien von den raw Ressourcen in den APK der SDCARD kopieren. Alternativ können Sie auch auf die Dateien direkt in den raw Ressourcen zugreifen. Diese letztere Methode wird empfohlen. Es
spart Zeit beim ersten Programm- laden und spart Platz auf der SDCARD.

Kopieren von Dateien auf die SDCARD

Bild, Audio und andere Dateien, die das Programm möglicherweise braucht, können in der .apk Datei in res.raw geliefert werden. Die Daten in diesen Dateien werden von res.raw kopiert und in Dateien in das Anwendungs- ”data” Unterverzeichnis geschrieben. Diese Kopierarbeit erfogt durch die loadGraphics() Methode.

Wenn Ihr Programm nicht unbedingt alle Dateien erstellen soll, können Sie einfach den Aufruf der loadGraphics()-Methode auskommentieren.



Wenn Sie Dateien haben, die Sie laden möchten, gehen Sie folgendermaßen vor:
Außerhalb von Eclipse, wählen Sie diese aus und kopieren Sie die Datei.
In Eclipse, mit einem Rechtsclick auf res.raw klicken und wählen Sie Paste (Einfügen)
Die Datei wird in res.raw. mit dem kopierten Dateinamen und der Erweiterung eingefügt werden.
Sobald die Dateien umbenannt und in res.raw geladen wurden, ändern Sie den Code in LoadGraphics. Für jede Datei müssen Sie den Dateinamen der res.raw Quelle im InputStream - Codezeile und den Zieldateinamen in der Codezeile PathA ändern. Sie können ausschneiden und einfügen für ähnliche Codezeilen verwenden, wenn Sie mehr als drei Dateien zu laden haben.
Hinweis: Benennen Sie die Dateien nicht mit den Erweiterungen. Lassen die Erweiterungen weg.
Hinweis: Wenn Sie sich entscheiden, die res.raw Dateien, Cartman, Boing und whee, zu löschen, müssen Sie die Verweise darauf in den LoadGraphics-Code auskommentieren.
Vergessen Sie nicht, Ihre Änderungen zu speichern.
Der Code prüft, ob jede Datei vorhanden ist, bevor Sie geladen werden. Wenn dann die Datei existiert wird sie nicht neu geladen werden. Dadurch wird die Ladezeit der nachfolgenden Ausführungen des APK reduziert.
Wenn die Dateien geladen werden, werden Punkte auf dem Bildschirm angezeigt, um

den Fortschritt zu melden. Nach jeden dreißig Zeilen aus Punkten, wird die Meldung ”Continuing load(N)” angezeigt. N ist eine Zahl, die jedes Mal neu inkrementiert wird, wenn die Meldung ”Continuing load(N)” angezeigt wird. Wenn Sie den letzten Wert ”Continuing load(N)” beobachten, können Sie das nutzen um die Anzahl N in einen Prozentwert zu verändern. Suchen Sie sich hierzu die Methode.
”onProgressUpdate” bei Zeile 201 in Basic.java. Ändern Sie den Wert des MAX_UPDATE_COUNT zu den zuletzt beobachteten Wert der N + 1

Verlassen, Dateien in der APK

Wenn Sie die Dateien nicht auf die SDCARD kopieren, sondern nur das res.raw wie oben gezeigt nutzen, müssen Sie Ihr BASIC! Programm nicht ändern, denn Sie haben Zugriff auf die Dateien aus dem APK. Die Tatsache, die Sie von einem APK ausführen bewirkt, dass sie Zugriff auf das APK haben.
Die Funktion zum Laden von Dateien direkt aus der APK ist für die folgenden Befehle implementiert:
Gr.Bitmap.Load
Audio.Load
SP.Load
Text.Open
Byte.Open

Kopieren Ihres BASIC! Programms in das APK

Irgendwie, kopieren Sie den Quellcode für Ihr BASIC! Programm in die Zwischenablage. In res.raw, doppelklicken Sie auf my_program.bas. Es öffnet sich das Fenster auf der rechten Seite. Klicken Sie auf das Fenster.
Wählen Sie im Menü Edit (Bearbeiten) Select All (Alles Auswählen) gefolgt von Past (Einfügen). Speichern Sie die Änderung.
Überprüfen Sie die Registerkarte Probleme am Ende von Eclipse, stellen Sie sicher, dass Sie keine Codierung Fehler in Basic.java gemacht haben.

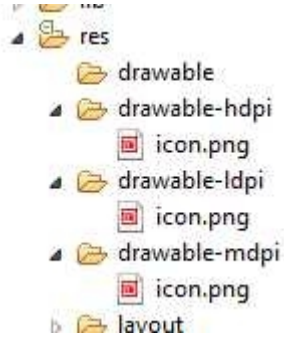
Application ICONs (Anwendungssymbole)

Android legt fest, dass Anwendungs-Symbole in drei spezifischen Größen zur Verfügung gestellt werden müssen: niedrige dpi (36x36 Pixel), mittlere dpi (48x48 Pixel) und hohe dpi (72x72 Pixel).
Die Symbole müssen auch .png Dateien sein.
There are tens of thousands of free icons available on the web.

Es sind Zehntausende von freien Icons im Web verfügbar.

Die meisten Bildbearbeitungs-Programme können verwendet werden, um zu probieren die Symbole in die richtigen Größen und in
PNG-Dateien umzuwandeln.

Wenn Sie Ihre Anwendung nicht auf dem Android Markt bringen wollen, dann müssen Sie sich nicht wirklich Sorgen um das Einhalten des exakt Richtigen machen.
Um Ihr Symbol in Ihre Anwendung in res, zu bekommen, muss es offene drawable-LDPI, drawable-MDPI, drawable-hdpi sein.

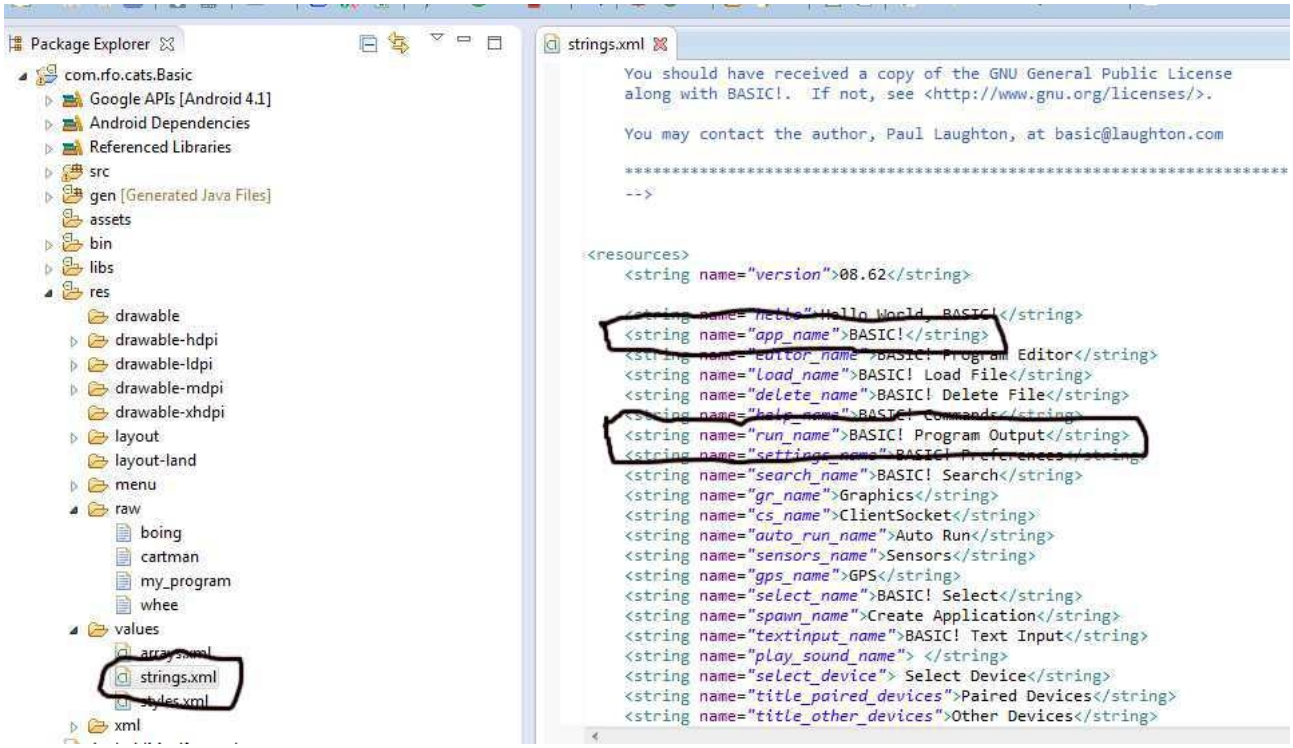


Für jede der Icon-Größen:
Außerhalb von Eclipse, kopieren Sie die Icon-Datei.
In Eclipse, mit Rechtsclick auf die entsprechende kopierte drawable-Icon Größe klicken und mit Past (Einfügen).
Mit rechter Maustaste auf das Symbol der icon.png Datei klicken und diese löschen.
Wählen Sie das neu eingefügte Symbol und benennt sie es um in "icon.png ", indem Sie Datei -› Rename (Umbenennen) benutzen.

Ja, es ist eine mühsame Arbeit.

Benennung der Anwendung

Als nächstes werden wir den Namen der Anwendung definieren und die Bezeichnung setzen, die auf der Textausgabe des Bildschirms angezeigt wird.
Unter res, öffnen Sie "values" (Werte). Doppelklicken Sie auf strings.xml um es zu öffnen.
Wenn Sie nicht sehen, was unten dargestellt ist, auf den "string.xml" Reiter im string.xml Fenster clicken.



Der Name der Anwendung wird durch den ”app_name”-Parameter definiert. In diesem Fall ändern Sie ”BASIC!” in ”CATS!”

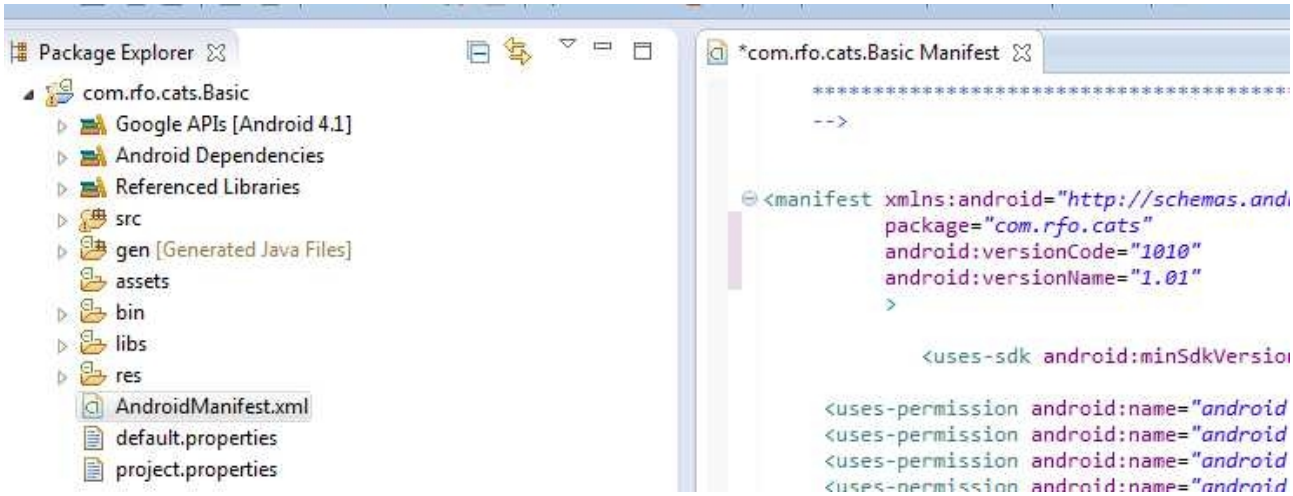
Der Titel in der Textausgabebildschirm- Titelleiste wird vom ”run_name”-Parameter definiert. In diesem Fall ändern Sie ”BASIC!” in ”CATS!”

Nehmen Sie die Änderungen vor, dann auf das X der Registerkarte klicken, um zu schließen und die Änderungen speichern.

Einstellung der Version Nummer und des Namens

Wenn Sie vorhaben, die Anwendung auf dem Android Market zu stellen, müssen Sie die Versionsnummer und Namen für jede neue Version wechseln

Ändern Sie den Versions- Informationen durch einen Doppelklick auf der AndroidManifest.xml-Datei.



Nehmen Sie die entsprechenden Änderungen am Android: versionCode und am Android: versionName vor, klicken Sie auf das X in der Registerkarte um zu schließen und die Änderungen speichern. Wenn Sie die BASIC! Version$()-Funktion verwenden möchten, damit Programm Ihre Versionsnummer liest, können Sie auch die Version Nummer ändern in res › values - › strings.

Berechtigungen

BASIC! nutzt viele Features, wegen denen der APK Benutzer gewarnt wird und zustimmen muss. Ihre besondere APK braucht vielleicht nicht alle oder nur einen Teil dieser Berechtigungen. Diese Erlaubnis- Benachrichtigungen werden in dem AndroidManifest.xml festgehalten.
Die Erlaubnis Meldungen können wie folgt aussehen:

‹uses-permission android.name=”.....

Schauen Sie sich diese an.
Wenn Sie der Meinung sind, dass Ihre APK diese nicht braucht, löschen Sie diese oder kommentieren Sie sie aus.
Bitte halten Sie die Berechtigung für”Vibrieren”. Wenn Sie diese Berechtigung nicht erteilen, kann das APK abstürzen, wenn es beendet wird. Diese Berechtigung sieht so aus wie:
‹uses-permission android:name=”android.permission.VIBRATE” android:required=”false”›‹/uses-permission›

Wenn Ihre Anwendung die SDCARD verwendet, dürfen Sie ‹uses-permission android:name=”android.permission.WRITE_EXTERNAL_STORAGE” /›nicht heraus kommentieren.

Achten Sie darauf, Ihr APK nach Änderungen an der Berechtigung zu testen.

Starten nach dem Booten des Gerätes

Sie APK können so eingerichtet werden, dass erst automatisch starten wenn das Android-Gerät gebootet wurde. Dies wird erreicht, indem Sie einen Parameter in der AndroidManifest.xml ändern.

Finden Sie die Codezeile (etwa Zeile 72):

‹receiver android:enabled=”false” android:name=”.BootUpReceiver”›

und ändern Sie es in:

‹receiver android:enabled=”true” android:name=”.BootUpReceiver”›

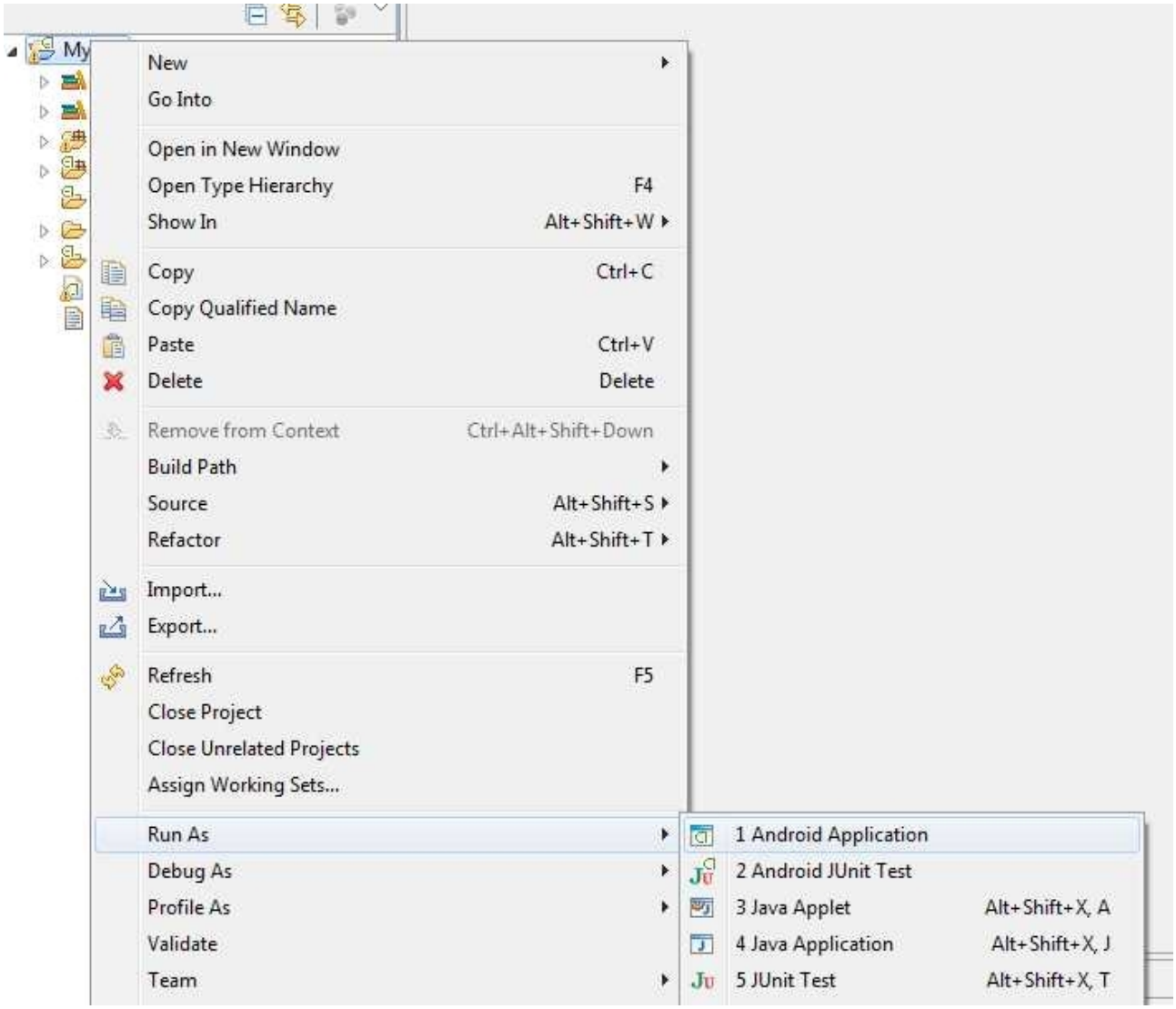
Kompilieren und Testen der Anwendung

Stellen Sie sicher, dass Sie die USB-Treiber für Ihr Android-Gerät auf Ihrem Computer installiert haben. Schließen Sie das Gerät an den Computer per USB-Kabel an.

Auf dem Gerät, öffnen Sie die Anwendungs Einstellungen.

Wählen Sie Applications -› Development -› USB debugging (Anwendungen -› Entwicklung -› USB-Debugging).

In Eclipse, clicken Sie mit der rechten Maustaste auf ”com.rfo.cats.BASIC” und wählen Sie Run As -› Android Applikation



Wenn alles gut gegangen ist, sollte Ihre Anwendung kompiliert sein und Sie bekommen sie auf Ihr Gerät heruntergeladen und sie kann ausgeführt werden.

Erstellen einer APK-Datei

Jetzt ist alles, was noch zu tun ist, die Apk-Datei zu erstellen. Folgen Sie dem Prozess der verwendet wird um den Apk Cats! zu erstellen. Stellen Sie sicher, dass alle Dateien, die Sie geändert wurden, gespeichert wurden. Der Erstellungsprozess des APK speichert geänderte Dateien nicht automatisch.

Fertig

Nicht schlecht, oder ?

Anhang E - BASIC! Distribution License

BASIC! wird unter den Bedingungen der GNU GENERAL PUBLIC LICENSE, die hier wiedergegeben wird, verteilt.

Deutsche Übersetzung der Version 3, 29. Juni 2007

Den offiziellen englischen Originaltext finden Sie unter http://www.gnu.org/licenses/gpl.html.

Deutsche Übersetzung: Peter Gerwinski, 5.7.2007

Dies ist eine inoffizielle deutsche Übersetzung der GNU General Public License, die nicht von der Free Software Foundation herausgegeben wurde. Es handelt sich hierbei nicht um eine rechtsgültige Festlegung der Bedingungen für die Weitergabe von Software, die der GNU GPL unterliegt; dies leistet nur der englische Originaltext. Wir hoffen jedoch, daß diese Übersetzung deutschsprachigen Lesern helfen wird, die GNU GPL besser zu verstehen.

This is an unofficial translation of the GNU General Public License into German. It was not published by the Free Software Foundation, and does not legally state the distribution terms for software that uses the GNU GPLonly the original English text of the GNU GPL does that. However, we hope that this translation will help German speakers understand the GNU GPL better.

GNU General Public License

Deutsche Übersetzung der Version 3, 29. Juni 2007

Diese Übersetzung ist kein rechtskräftiger Ersatz für die englischsprachige Originalversion!

Vorwort

Die GNU General Public License – die Allgemeine Öffentliche GNU-Lizenz – ist eine freie Copyleft-Lizenz für Software und andere Arten von Werken.

Die meisten Lizenzen für Software und andere nutzbaren Werke sind daraufhin entworfen worden, Ihnen die Freiheit zu nehmen, die Werke mit anderen zu teilen

und zu verändern. Im Gegensatz dazu soll Ihnen die GNU General Public License die Freiheit garantieren, alle Versionen eines Programms zu teilen und zu verändern. Sie soll sicherstellen, daß die Software für alle ihre Benutzer frei bleibt. Wir, die Free Software Foundation, nutzen die GNU General Public License für den größten Teil unserer Software; sie gilt außerdem für jedes andere Werk, dessen Autoren es auf diese Weise freigegeben haben. Auch Sie können diese Lizenz auf Ihre Programme anwenden.

Wenn wir von freier Software sprechen, so beziehen wir uns auf Freiheit, nicht auf den Preis. Unsere Allgemeinen Öffentlichen Lizenzen sind darauf angelegt, sicherzustellen, daß Sie die Freiheit haben, Kopien freier Software zu verbreiten (und dafür etwas zu berechnen, wenn Sie möchten), die Möglichkeit, daß Sie die Software als Quelltext erhalten oder den Quelltext auf Wunsch bekommen, daß Sie die Software ändern oder Teile davon in neuen freien Programmen verwenden dürfen und daß Sie wissen, daß Sie dies alles tun dürfen.

Um Ihre Rechte zu schützen, müssen wir andere daran hindern, Ihnen diese Rechte zu verweigern oder Sie aufzufordern, auf diese Rechte zu verzichten. Aus diesem Grunde tragen Sie eine Verantwortung, wenn Sie Kopien der Software verbreiten oder die Software verändern: die Verantwortung, die Freiheit anderer zu respektieren.

Wenn Sie beispielsweise die Kopien eines solchen Programms verbreiten – kostenlos oder gegen Bezahlung – müssen Sie an die Empfänger dieselben Freiheiten weitergeben, die Sie selbst erhalten haben. Sie müssen sicherstellen, daß auch die Empfänger die Software im Quelltext erhalten bzw. den Quelltext erhalten können. Und Sie müssen ihnen diese Bedingungen zeigen, damit sie ihre Rechte kennen.

Software-Entwickler, die die GNU GPL nutzen, schützen Ihre Rechte in zwei Schritten: (1) Sie machen ihr Urheberrecht (Copyright) auf die Software geltend, und (2) sie bieten Ihnen diese Lizenz an, die Ihnen das Recht gibt, die Software zu vervielfältigen, zu verbreiten und/oder zu verändern.

Um die Entwickler und Autoren zu schützen, stellt die GPL darüberhinaus klar, daß für diese freie Software keinerlei Garantie besteht. Um sowohl der Anwender als auch der Autoren Willen erfordert die GPL, daß modifizierte Versionen der Software als solche gekennzeichnet werden, damit Probleme mit der modifizierten Software nicht fälschlicherweise mit den Autoren der Originalversion in Verbindung gebracht werden.

Manche Geräte sind daraufhin entworfen worden, ihren Anwendern zu verweigern, modifizierte Versionen der darauf laufenden Software zu installieren oder laufen zu lassen, wohingegen der Hersteller diese Möglichkeit hat. Dies ist grundsätzlich unvereinbar mit dem Ziel, die Freiheit der Anwender zu schützen, die Software zu modifizieren. Derartige gezielte mißbräuchliche Verhaltensmuster finden auf dem Gebiet persönlicher Gebrauchsgegenstände statt – also genau dort, wo sie am wenigsten akzeptabel sind. Aus diesem Grunde wurde diese Version der GPL daraufhin entworfen, diese Praxis für diese Produkte zu verbieten. Sollten derartige Probleme substantiell auf anderen Gebieten auftauchen, sind wir bereit, diese Regelung auf diese Gebiete auszudehnen, soweit dies notwendig ist, um die Freiheit der Benutzer zu schützen.

Schließlich und endlich ist jedes Computerprogramm permanent durch Software-Patente bedroht. Staaten sollten es nicht zulassen, daß Patente die Entwicklung und Anwendung von Software für allgemein einsetzbare Computer einschränken, aber in Staaten, wo dies geschieht, wollen wir die spezielle Gefahr vermeiden, daß Patente dazu verwendet werden, ein freies Programm im Endeffekt proprietär zu

machen. Um dies zu verhindern, stellt die GPL sicher, daß Patente nicht verwendet werden können, um das Programm nicht-frei zu machen.

Es folgen die präzisen Bedingungen für das Kopieren, Verbreiten und Modifizieren.
LIZENZBEDINGUNGEN
0. Definitionen

„Diese Lizenz" bezieht sich auf die Version 3 der GNU General Public License.

Mit „Urheberrecht" sind auch urheberrechtähnliche Rechte gemeint, die auf andere Arten von Werken Anwendung finden, beispielsweise auf Fotomasken in der Halbleitertechnologie.

„Das Programm" bezeichnet jedes urheberrechtlich schützbare Werk, das unter diese Lizenz gestellt wurde. Jeder Lizenznehmer wird als „Sie" angeredet. „Lizenznehmer" und „Empfänger" können natürliche oder rechtliche Personen sein.

Ein Werk zu „modifizieren" bedeutet, aus einem Werk zu kopieren oder es ganz oder teilweise auf eine Weise umzuarbeiten, die eine urheberrechtliche Erlaubnis erfordert und kein Eins-zu-eins-Kopieren darstellt. Das daraus hervorgehende Werk wird als „modifizierte Version" des früheren Werks oder als auf dem früheren Werk „basierendes" Werk bezeichnet.

Ein „betroffenes Werk" bezeichnet entweder das unmodifizierte Programm oder ein auf dem Programm basierendes Werk.

Ein Werk zu „propagieren" bezeichnet jedwede Handlung mit dem Werk, für die man, wenn unerlaubt begangen, wegen Verletzung anwendbaren Urheberrechts direkt oder indirekt zur Verantwortung gezogen würde, ausgenommen das Ausführen auf einem Computer oder das Modifizieren einer privaten Kopie. Unter das Propagieren eines Werks fallen Kopieren, Weitergeben (mit oder ohne Modifikationen), öffentliches Zugänglichmachen und in manchen Staaten noch weitere Tätigkeiten.

Ein Werk zu „übertragen" bezeichnet jede Art von Propagation, die es Dritten ermöglicht, das Werk zu kopieren oder Kopien zu erhalten. Reine Interaktion mit einem Benutzer über ein Computer-Netzwerk ohne Übergabe einer Kopie ist keine Übertragung.

Eine interaktive Benutzerschnittstelle zeigt „angemessene rechtliche Hinweise" in dem Umfang, daß sie eine zweckdienliches und deutlich sichtbare Funktion bereitstellt, die (1) einen angemessenen Copyright-Vermerk zeigt und (2) dem Benutzer mitteilt, daß keine Garantie für das Werk besteht (ausgenommen in dem Umfang, in dem Garantie gewährt wird), daß Lizenznehmer das Werk gemäß dieser Lizenz übertragen dürfen und wie man ein Exemplar dieser Lizenz zu Gesicht bekommen kann. Wenn die Benutzerschnittstelle eine Liste von Benutzerkommandos oder Optionen anzeigt, zum Beispiel ein Menü, dann erfüllt ein deutlich sichtbarer Punkt in dieser Liste dieses Kriterium.
1. Quelltext

Der „Quelltext" eines Werkes bezeichnet diejenige Form des Werkes, die für Bearbeitungen vorzugsweise verwendet wird. „Objekt-Code" bezeichnet jede Nicht-Quelltext-Form eines Werks.

Eine „Standardschnittstelle" bezeichnet eine Schnittstelle, die entweder ein offizieller Standard eines anerkannten Standardisierungsgremiums ist oder – im Falle von Schnittstellen, die für eine spezielle Programmiersprache spezifiziert wurden – eine Schnittstelle, die unter Entwicklern, die in dieser

Programmiersprache arbeiten, weithin gebräuchlich ist.

Die „Systembibliotheken" eines ausführbaren Werks enthalten alles, ausgenommen das Werk als Ganzes, was (a) normalerweise zum Lieferumfang einer Hauptkomponente gehört, aber selbst nicht die Hauptkomponente ist, und (b) ausschließlich dazu dient, das Werk zusammen mit der Hauptkomponente benutzen zu können oder eine Standardschnittstelle zu implementieren, für die eine Implementation als Quelltext öffentlich erhältlich ist. Eine „Hauptkomponente" bezeichnet in diesem Zusammenhang eine größere wesentliche Komponente (Betriebssystemkern, Fenstersystem usw.) des spezifischen Betriebssystems (soweit vorhanden), auf dem das ausführbare Werk läuft, oder des Compilers, der zur Erzeugung des Objekt-Codes eingesetzt wurde, oder des für die Ausführung verwendeten Objekt-Code-Interpreters.

Der „korrespondierende Quelltext" eines Werks in Form von Objekt-Code bezeichnet den vollständigen Quelltext, der benötigt wird, um das Werk zu erzeugen, es zu installieren, um (im Falle eines ausführbaren Werks) den Objekt-Code auszuführen und um das Werk zu modifizieren, einschließlich der Skripte zur Steuerung dieser Aktivitäten. Er schließt jedoch nicht die Systembibliotheken, allgemein einsetzbare Werkzeuge oder allgemein erhältliche freie Computerprogramme mit ein, die in unmodifizierter Form verwendet werden, um die o.a. Tätigkeiten durchzuführen, die aber nicht Teil des Werks sind. Zum Beispiel enthält der korrespondierende Quelltext die zum Programmquelltext gehörenden Schnittstellendefinitionsdateien sowie die Quelltexte von dynamisch eingebundenen Bibliotheken und Unterprogrammen, auf die das Werk konstruktionsbedingt angewiesen ist, beispielsweise durch komplexe Datenkommunikation oder Ablaufsteuerung zwischen diesen Unterprogrammen und anderen Teilen des Werks.

Der korrespondierende Quelltext braucht nichts zu enthalten, das der Anwender aus anderen Teilen des korrespondierenden Quelltextes automatisch regenerieren kann.

Der korrespondierende Quelltext eines Werks in Quelltextform ist das Werk selbst.
2. Grundlegende Genehmigungen

Alle unter dieser Lizenz gewährten Rechte werden gewährt auf Grundlage des Urheberrechts an dem Programm, und sie sind unwiderruflich, solange die festgelegten Bedingungen erfüllt sind. Diese Lizenz erklärt ausdrücklich Ihr uneingeschränktes Recht zur Ausführung des unmodifizierten Programms. Die beim Ausführen eines betroffenen Werks erzeugten Ausgabedaten fallen unter diese Lizenz nur dann, wenn sie, in Anbetracht ihres Inhalts, ein betroffenes Werk darstellen. Diese Lizenz erkennt Ihr im Urheberrecht vorgesehenes Recht auf angemessene Benutzung – oder seine Entsprechung – an.

Sie dürfen betroffene Werke, die Sie nicht übertragen, uneingeschränkt erzeugen, ausführen und propagieren, solange Ihre Lizenz ansonsten in Kraft bleibt. Sie dürfen betroffene Werke an Dritte übertragen für den einzigen Zweck, Modifikationen exklusiv für Sie durchzuführen oder Einrichtungen für Sie bereitzustellen, um diese Werke auszuführen, vorausgesetzt, Sie erfüllen alle Bedingungen dieser Lizenz für das Übertragen von Material, dessen Urheberrecht nicht bei Ihnen liegt. Diejenigen, die auf diese Weise betroffene Werke für Sie anfertigen oder ausführen, müssen dies ausschließlich in Ihrem Namen tun, unter Ihrer Anleitung und Kontrolle und unter Bedingungen, die ihnen verbieten, außerhalb ihrer Beziehung zu Ihnen weitere Kopien Ihres urheberrechtlich geschützten Materials anzufertigen.

Übertragung ist in jedem Fall ausschließlich unter den unten aufgeführten Bedingungen gestattet. Unterlizensierung ist nicht gestattet, ist aber wegen §10 unnötig.

3. Schutz von Anwenderrechten vor Umgehungsverbotgesetzen

Kein betroffenes Werk darf als Teil eines wirksamen technischen Mechanismus' unter jedwedem anwendbarem Recht betrachtet werden, das die Auflagen von Artikel 11 des am 20. Dezember 1996 verabschiedeten WIPO-Urheberrechtsvertrags oder unter vergleichbaren Gesetzen, die die Umgehung derartiger Mechanismen verbietet oder einschränkt.

Wenn Sie ein betroffenes Werk übertragen, verzichten Sie auf jedes Recht, die Umgehung technischer Mechanismen zu verbieten, insoweit diese Umgehung durch die Ausübung der von dieser Lizenz gewährten Rechte in bezug auf das betroffene Werk herbeigeführt wird, und Sie weisen jede Absicht von sich, die Benutzung oder Modifikation des Werks zu beschränken, um Ihre Rechtsansprüche oder Rechtsansprüche Dritter zum Verbot der Umgehung technischer Mechanismen gegen die Anwender des Werks durchzusetzen.

4. Unveränderte Kopien

Sie dürfen auf beliebigen Medien unveränderte Kopien des Quelltextes des Programms, wie sie ihn erhalten, übertragen, sofern Sie auf deutliche und angemessene Weise auf jeder Kopie einen angemessenen Urheberrechts-Vermerk veröffentlichen, alle Hinweise intakt lassen, daß diese Lizenz und sämtliche gemäß §7 hinzugefügten Einschränkungen auf den Quelltext anwendbar sind, alle Hinweise auf das Nichtvorhandensein einer Garantie intakt lassen und allen Empfängern gemeinsam mit dem Programm ein Exemplar dieser Lizenz zukommen lassen.

Sie dürfen für jede übertragene Kopie ein Entgelt – oder auch kein Entgelt – verlangen, und Sie dürfen Kundendienst- oder Garantieleistungen gegen Entgelt anbieten.

5. Übertragung modifizierter Quelltextversionen

Sie dürfen ein auf dem Programm basierendes Werk oder die nötigen Modifikationen, um es aus dem Programm zu generieren, kopieren und übertragen in Form von Quelltext unter den Bestimmungen von §4, vorausgesetzt, daß Sie zusätzlich alle im folgenden genannten Bedingungen erfüllen:

a) Das veränderte Werk muß auffällige Vermerke tragen, die besagen, daß Sie es modifiziert haben, und die ein darauf bezogenes Datum angeben.
b) Das veränderte Werk muß auffällige Vermerke tragen, die besagen, daß es unter dieser Lizenz einschließlich der gemäß §7 hinzugefügten Bedingungen herausgegeben wird. Diese Anforderung wandelt die Anforderung aus §4 ab, „alle Hinweise intakt zu lassen".
c) Sie müssen das Gesamtwerk als Ganzes gemäß dieser Lizenz an jeden lizensieren, der in den Besitz einer Kopie gelangt. Diese Lizenz wird daher – ggf. einschließlich zusätzlicher Bedingungen gemäß §7 – für das Werk als Ganzes und alle seine Teile gelten, unabhängig davon, wie diese zusammengepackt werden. Diese Lizenz erteilt keine Erlaubnis, das Werk in irgendeiner anderen Weise zu lizensieren, setzt aber eine derartige Erlaubnis nicht außer Kraft, wenn Sie sie diese gesondert erhalten haben.
d) Wenn das Werk über interaktive Benutzerschnittstellen verfügt, müssen diese jeweils angemessene rechtliche Hinweise anzeigen. Wenn allerdings das Programm interaktive Benutzerschnittstellen hat, die keine angemessenen rechtlichen Hinweise anzeigen, braucht Ihr Werk nicht dafür zu sorgen, daß sie dies tun.

Die Zusammenstellung eines betroffenen Werks mit anderen gesonderten und unabhängigen Werken, die nicht ihrer Natur nach Erweiterungen des betroffenen Werks sind und die nicht mit ihm in einer Weise kombiniert sind, um ein größeres Programm zu bilden, in oder auf einem Speicher- oder Verbreitungsmedium wird als „Aggregat" bezeichnet, wenn die Zusammenstellung und das sich für sie ergebende Urheberrecht nicht dazu verwendet werden, den Zugriff oder die Rechte der Benutzer der Zusammenstellung weiter einzuschränken, als dies die einzelnen Werke erlauben. Die Aufnahme des betroffenen Werks in ein Aggregat sorgt nicht dafür, daß diese Lizenz auf die anderen Teile des Aggregats wirke.
6. Übertragung in Nicht-Quelltext-Form

Sie dürfen ein betroffenes Werk in Form von Objekt-Code unter den Bedingungen der Paragraphen 4 und 5 kopieren und übertragen – vorausgesetzt, daß Sie außerdem den maschinenlesbaren korrespondierenden Quelltext unter den Bedingungen dieser Lizenz übertragen auf eine der folgenden Weisen:

a) Sie übertragen den Objekt-Code in einem physikalischen Produkt (einschließlich ein physikalisches Speichermedium) gemeinsam mit dem korrespondierenden Quelltext, der sich unveränderlich auf einem haltbaren physikalischen Medium befindet, das üblicherweise für den Austausch von Software verwendet wird.
b) Sie übertragen den Objekt-Code in einem physikalischen Produkt (einschließlich ein physikalisches Speichermedium) gemeinsam mit einem schriftlichen Angebot, das mindestens drei Jahre lang gültig sein muß und so lange, wie Sie Ersatzteile und Kundendienst für dieses Produktmodell anbieten, jedem, der im Besitz des Objekt-Codes ist, entweder (1) eine Kopie des korrespondierenden Quelltextes der gesamten Software, die in dem Produkt enthalten und von dieser Lizenz betroffen ist, zur Verfügung zu stellen – auf einem haltbaren physikalischen Medium, das üblicherweise für den Austausch von Software verwendet wird, und zu nicht höheren Kosten als denen, die begründbar durch den physikalischen Vorgang der Übertragung des Quelltextes anfallen, oder (2) kostenlosen Zugriff, um den korrespondierenden Quelltext von einem Netzwerk-Server zu kopieren.
c) Sie übertragen Kopien des Objekt-Codes gemeinsam mit einer Kopie des schriftlichen Angebots, den korrespondierenden Quelltext zur Verfügung zu stellen. Diese Alternative ist nur für gelegentliche, nicht-kommerzielle Übertragung zulässig und nur, wenn Sie den Objekt-Code als mit einem entsprechenden Angebot gemäß Absatz 6b erhalten haben.
d)

Sie übertragen den Objekt-Code dadurch, daß Sie Zugriff auf eine dafür vorgesehene Stelle gewähren, und bieten gleichwertigen Zugriff auf den korrespondierenden Quelltext auf gleichem Weg auf dieselbe Stelle und ohne zusätzliche Kosten. Sie müssen nicht von den Empfängern verlangen, den korrespondierenden Quelltext gemeinsam mit dem Objekt-Code zu kopieren. Wenn es sich bei der für das Kopieren vorgesehenen Stelle um einen Netzwerk-Server handelt, darf sich der korrespondierende Quelltext auf einem anderen Server befinden (von Ihnen oder von einem Dritten betrieben), der gleichwertige Kopiermöglichkeiten unterstützt – vorausgesetzt Sie legen dem Objekt-Code klare Anleitungen bei, die besagen, wo der korrespondierende Quelltext zu finden ist. Unabhängig davon, welcher Netzwerk-Server den korrespondierenden Quelltext beherbergt, bleiben Sie verpflichtet, sicherzustellen, daß dieser lange genug bereitgestellt wird, um diesen Bedingungen zu genügen.
e) Sie übertragen den Objekt-Code unter Verwendung von Peer-To-Peer-Übertragung – vorausgesetzt, Sie informieren andere Teilnehmer darüber, wo der Objekt-Code und der korrespondierende Quelltext des Werks unter den Bedingungen von Absatz 6d öffentlich und kostenfrei angeboten werden.

Ein abtrennbarer Anteil des Objekt-Codes, dessen Quelltext von dem korrespondierenden Quelltext als Systembibliothek ausgeschlossen ist, braucht bei der Übertragung des Werks als Objekt-Code nicht miteinbezogen zu werden.

Ein „Benutzerprodukt" ist entweder (1) ein „Endbenutzerprodukt", womit ein materieller persönlicher Besitz gemeint ist, der normalerweise für den persönlichen oder familiären Gebrauch oder im Haushalt eingesetzt wird, oder (2) alles, was für den Einbau in eine Wohnung hin entworfen oder dafür verkauft wird. Bei der Entscheidung, ob ein Produkt ein Endbenutzerprodukt ist, sollen Zweifelsfälle als erfaßt gelten. Wenn ein spezieller Anwender ein spezielles Produkt erhält, bezeichnet „normalerweise einsetzen" eine typische oder weitverbreitete Anwendung dieser Produktklasse, unabhängig vom Status des speziellen Anwenders oder der Art und Weise, wie der spezielle Anwender des spezielle Produkt tatsächlich einsetzt oder wie von ihm erwartet wird, daß er es einsetzt. Ein Produkt gilt als Endbenutzerprodukt unabhängig davon, ob es substantiellen kommerziellen, industriellen oder nicht-endbenutzerspezifischen Nutzen hat, es sei denn, dieser Nutzen stellt das einzige signifikante Anwendungsgebiet des Produkts dar.

Mit „Installationsinformationen" für ein Benutzerprodukt sind jedwede Methoden, Prozeduren, Berechtigungsschlüssel oder andere informationen gemeint, die notwendig sind, um modifizierte Versionen eines betroffenen Werks, die aus einer modifizierten Version seines korrespondierenden Quelltextes hervorgegangen sind, auf dem Produkt zu installieren und auszuführen. Die Informationen müssen ausreichen, um sicherzustellen, daß das Weiterfunktionieren des modifizierten Objekt-Codes in keinem Fall verhindert oder gestört wird aus dem einzigen Grunde, weil Modifikationen vorgenommen worden sind.

Wenn Sie Objekt-Code gemäß diesem Paragraphen innerhalb oder zusammen mit oder speziell für den Gebrauch innerhalb eines Benutzerprodukts übertragen und die Übertragung als Teil einer Transaktion stattfindet, in der das Recht auf den Besitz und die Benutzung des Benutzerprodukts dauerhaft auf den Empfänger übergeht (unabhängig davon, wie diese Transaktion charakterisiert ist), müssen dem gemäß diesem Paragraphen mitübertragenen korrespondierenden Quelltext die Installationsinformationen beiliegen. Diese Anforderung gilt jedoch nicht, wenn weder Sie noch irgendeine Drittpartei die Möglichkeit behält, modifizierten Objekt-Code auf dem Benutzerprodukt zu installieren (zum Beispiel, wenn das Werk in einem ROM installiert wurde).

Die Anforderung, Installationsinformationen bereitzustellen, schließt keine Anforderung mit ein, weiterhin Kundendienst, Garantie oder Updates für ein Werk bereitzustellen, das vom Empfänger modifiziert oder installiert worden ist, oder für das Benutzerprodukt, in dem das Werk modifiziert oder installiert worden ist. Der Zugriff auf ein Computer-Netzwerk darf verweigert werden, wenn die Modifikation selbst die Funktion des Netzwerks grundlegend nachteilig beeinflußt oder wenn sie die Regeln und Protokolle für die Kommunikation über das Netzwerk verletzt.

Der korrespondierende Quelltext und die Installationsinformationen, die in Übereinstimmung mit diesem Paragraphen übertragen werden, müssen in einem öffentlich dokumentierten Format vorliegen (für das eine Implementation in Form von Quelltext öffentlich zugänglich ist), und sie dürfen keine speziellen Passwörter oder Schlüssel für das Auspacken, Lesen oder Kopieren erfordern.
7. Zusätzliche Bedingungen

„Zusätzliche Genehmigungen" sind Bedingungen, die die Bedingungen dieser Lizenz ergänzen, indem sie Ausnahmen von einer oder mehreren Auflagen zulassen. Zusätzliche Genehmigungen zur Anwendung auf das gesamte Programm sollen so

betrachtet werden, als wären sie in dieser Lizenz enthalten, soweit dies unter anwendbarem Recht zulässig ist. Wenn zusätzliche Genehmigungen nur für einen Teil des Programms gelten, darf dieser Teil separat unter diesen Genehmigungen verwendet werden; das gesamte Programm jedoch unterliegt weiterhin dieser Lizenz ohne Beachtung der zusätzlichen Genehmigungen.

Wenn Sie eine Kopie eines betroffenen Werks übertragen, dürfen Sie, wenn Sie es wünschen, jegliche zusätzliche Genehmigungen von dieser Kopie oder jedem Teil der Kopie entfernen. (Zusätzliche Genehmigungen dürfen so verfaßt sein, daß sie in bestimmten Fällen, wenn Sie das Werk modifizieren, entfernt werden müssen.) Sie dürfen Material, das Sie einem betroffenen Werk hinzufügen und für das Sie das Urheberrecht besitzen oder in entsprechender Form gewähren dürfen, mit zusätzlichen Genehmigungen ausstatten.

Ungeachtet jeglicher anderer Regelungen dieser Lizenz dürfen Sie für Material, das Sie einem betroffenen Werk hinzufügen (sofern Sie durch die Urheberrechtsinhaber dieses Materials autorisiert sind), die Bedingungen dieser Lizenz um folgendes ergänzen:

a) Gewährleistungsausschluß oder Haftungsbegrenzung abweichend von §§15 und 16 dieser Lizenz oder
b) die Anforderung, spezifizierte sinnvolle rechtliche Hinweise oder Autorenschaftshinweise in diesem Material oder in den angemessenen rechtlichen Hinweisen, die von den sie enthaltenen Werken angezeigt werden, zu erhalten, oder
c) das Verbot, die Herkunft des Materials falsch darzustellen oder die Anforderung, daß modifizierte Versionen des Materials auf angemessens Weise als vom Original verschieden markiert werden, oder
d) Begrenzung der Verwendung der Namen von Lizenzgebern oder Autoren des Materials für Werbezwecke oder
e) das Zurückweisen der Einräumung von Rechten gemäß dem Markenrecht zur Benutzung gewisser Produktnamen, Produkt- oder Service-Marken oder
f) die Erfordernis der Freistellung des Lizenznehmers und der Autoren des Materials durch jeden, der die Software (oder modifizierte Versionen davon) überträgt, mit vertraglichen Prämissen der Verantwortung gegenüber dem Empfänger für jede Verantwortung, die diese vertraglichen Prämissen diesen Lizenzgebern und Autoren direkt auferlegen.

Alle anderen hinzugefügten einschränkenden Bedingungen werden als „zusätzliche Einschränkungen" im Sinne von §10 betrachtet. Wenn das Programm, wie Sie es erhalten haben, oder ein Teil davon dieser Lizenz untersteht zuzüglich einer weiteren Bedingung, die eine zusätzliche Einschränkung darstellt, dürfen Sie diese Bedingung entfernen. Wenn ein Lizenzdokument eine zusätzliche Einschränkung enthält, aber die Relizensierung unter dieser Lizenz erlaubt, dürfen Sie dem betroffenen Werk Material hinzufügen, das den Bedingungen jenes Lizenzdokuments unterliegt, unter der Voraussetzung, daß die zusätzlichen Einschränkungen bei einer derartigen Relizensierung oder Übertragung verfallen.

Wenn Sie einem betroffenen Werk in Übereinstimmung mit diesem Paragraphen Bedingungen hinzufügen, müssen Sie in den betroffenen Quelltextdateien eine Aufstellung der zusätzlichen Bedingungen plazieren, die auf diese Quelltextdatei Anwendung finden, oder einen Hinweis darauf, wo die Zusätzlichen Bedingungen zu finden sind.

Zusätzliche Bedingungen, seien es Genehmigungen oder Einschränkungen, dürfen in Form einer separaten schriftlichen Lizenz oder in Form von Ausnahmen festgelegt werden; die o.a. Anforderungen gelten in jedem Fall.
8. Kündigung

Sie dürfen das Programm nicht verbreiten oder modifizieren, sofern es nicht durch diese Lizenz ausdrücklich gestattet ist. Jeder anderweitige Versuch der Verbreitung oder Modifizierung ist nichtig und beendet automatisch Ihre Rechte unter dieser Lizenz (einschließlich aller Patentlizenzen gemäß §11 Abs. 3).

Wenn Sie jedoch alle Verletzungen dieser Lizenz beenden, wird Ihre Lizenz durch einen speziellen Urheberrechtsinhaber wiederhergestellt, und zwar (a) vorübergehend, solange nicht bzw. bis der Rechteinhaber Ihre Lizenz ausdrücklich und endgültig kündigt, und (b) dauerhaft, sofern es der Rechteinhaber versäumt, Sie auf sinnvolle Weise auf die Lizenzverletzung innerhalb von 60 Tagen ab deren Beendigung hinzuweisen.

Darüberhinaus wird Ihre Lizenz durch einen speziellen Urheberrechtsinhaber permanent wiederhergestellt, wenn Sie der Rechteinhaber auf sinnvolle Weise auf die Verletzung hinweist, wenn außerdem dies das erste Mal ist, daß Sie auf die Verletzung dieser Lizenz (für jedes Werk) des Rechteinhabers hingewiesen werden, und wenn Sie die Verletzung innerhalb von 30 Tagen ab dem Eingang des Hinweises einstellen.

Die Beendigung Ihrer Rechte unter dieser Lizenz beendet nicht die Lizenzen Dritter, die von Ihnen Kopien oder Rechte unter dieser Lizenz erhalten haben. Wenn Ihre Rechte beendet und nicht dauerhaft wiederhergestellt worden sind, sind Sie nicht berechtigt, neue Lizenzen für dasselbe Material gemäß §10 zu erhalten.
9. Annahme der Lizenz keine Voraussetzung für den Besitz von Kopien

Um eine Kopie des Programms auszuführen, ist es nicht erforderlich, daß Sie diese Lizenz annehmen. Die nebenbei stattfindende Verbreitung eines betroffenen Werks, die sich ausschließlich als Konsequenz der Teilnahme an einer Peer-To-Peer-Datenübertragung ergibt, um eine Kopie entgegennehmen zu können, erfordert ebenfalls keine Annahme dieser Lizenz. Jedoch gibt Ihnen nichts außer dieser Lizenz die Erlaubnis, das Programm oder jedes betroffene Werk zu verbreiten oder zu verändern. Diese Handlungen verstoßen gegen das Urheberrecht, wenn Sie diese Lizenz nicht anerkennen. Indem Sie daher ein betroffenes Werk verändern oder propagieren, erklären Sie Ihr Einverständnis mit dieser Lizenz, die Ihnen diese Tätigkeiten erlaubt.
10. Automatische Lizensierung nachgeordneter Anwender

Jedesmal, wenn Sie ein betroffenes Werk übertragen, erhält der Empfänger automatisch vom ursprünglichen Lizenzgeber die Lizenz, das Werk auszuführen, zu verändern und zu propagieren – in Übereinstimmung mit dieser Lizenz. Sie sind nicht dafür verantwortlich, die Einhaltung dieser Lizenz durch Dritte durchzusetzen.

Eine „Organisations-Transaktion" ist entweder eine Transaktion, bei der die Kontrolle über eine Organisation oder das im wesentlichen gesamte Kapital einer solchen, übertragen wird, oder sie ist die Aufteilung einer Organisation in mehrere oder die Fusion mehrerer Organisationen zu einer. Wenn die Propagation eines betroffenen Werks durch eine Organisations-Transaktion erfolgt, erhält jeder an der Transaktion Beteiligte, der eine Kopie des Werks erhält, zugleich jedwede Lizenz an dem Werk, die der Interessenvorgänger des Beteiligten hatte, sowie das Recht auf den Besitz des korrespondierenden Quelltextes des Werks vom Interessenvorgänger, wenn dieser ihn hat oder mit vertretbarem Aufwand beschaffen kann.

Sie dürfen keine zusätzlichen Einschränkungen bzgl. der Ausübung der unter dieser Lizenz gewährten oder zugesicherten Rechte vornehmen. Beispielsweise dürfen Sie keine Lizenzgebühr oder sonstige Gebühr für die Ausübung der unter

dieser Lizenz gewährten Rechte verlangen, und Sie dürfen keine Rechtsstreitigkeit beginnen (eingeschlossen Kreuz- oder Gegenansprüche in einem Gerichtsverfahren), in der Sie unterstellen, daß irgendein Patentanspruch durch Erzeugung, Anwendung, Verkauf, Verkaufsangebot oder Import des Programms oder irgendeines Teils davon verletzt wurde.
11. Patente

Ein „Kontributor" ist ein Urheberrechtsinhaber, der die Benutzung des Programms oder eines auf dem Programm basierenden Werks unter dieser Lizenz erlaubt. Das auf diese Weise lizensierte Werk bezeichnen wir als die „Kontributor-Version" des Kontributors.

Die „wesentlichen Patentansprüche" eines Kontributors sind all diejenigen Patentansprüche, die der Kontributor besitzt oder kontrolliert, ob bereits erworben oder erst in Zukunft zu erwerben, die durch irgendeine Weise des gemäß dieser Lizenz erlaubten Erzeugens, Ausführens oder Verkaufens seiner Kontributor-Version verletzt würden. Dies schließt keine Patentansprüche ein, die erst als Konsequenz weiterer Modifizierung seiner Kontributor-Version entstünden. Für den Zweck dieser Definition schließt "Kontrolle" das Recht mit ein, Unterlizenzen für ein Patent zu erteilen auf eine Weise, die mit den Erfordernissen dieser Lizenz vereinbar ist.

Jeder Kontributor gewährt Ihnen eine nicht-exklusive, weltweite und gebührenfreie Patentlizenz gemäß den wesentlichen Patentansprüchen des Kontributors, den Inhalt seiner Kontributor-Version zu erzeugen, zu verkaufen, zum Verkauf anzubieten, zu importieren und außerdem auszuführen, zu modifizieren und zu propagieren.

In den folgenden drei Absätzen ist eine „Patentlizenz" jedwede ausdrückliche Vereinbarung oder Verpflichtung, wie auch immer benannt, ein Patent nicht geltend zu machen (beispielsweise eine ausdrückliche Erlaubnis, ein Patent zu nutzen oder eine Zusicherung, bei Patentverletzung nicht zu klagen). Jemandem eine solche Patentlizenz zu „erteilen" bedeutet, eine solche Vereinbarung oder Verpflichtung zu beschließen, ein Patent nicht gegen ihn durchzusetzen.

Wenn Sie ein betroffenes Werk übertragen, das wissentlich auf eine Patentlizenz angewiesen ist, und wenn der korrespondierende Quelltext nicht für jeden zum Kopieren zur Verfügung gestellt wird – kostenlos, unter den Bedingungen dieser Lizenz und über einen öffentlich zugänglichen Netzwerk-Server oder andere leicht zugängliche Mittel –, dann müssen Sie entweder (1) dafür sorgen, daß der korrespondierende Quelltext auf diese Weise verfügbar gemacht wird oder (2) dafür sorgen, daß Ihnen selbst die Vorteile der Patentlizenz für dieses spezielle Werk entzogen werden oder (3) in einer mit den Erfordernissen dieser Lizenz vereinbaren Weise bewirken, daß die Patentlizenz auf nachgeordnete Empfänger ausgedehnt wird. „Wissentlich angewiesen sein" bedeutet, daß Sie tatsächliches Wissen darüber haben, daß – außer wegen der Patentlizenz – Ihre Übertragung des betroffenen Werks in einen Staat oder die Benutzung des betroffenen Werks durch Ihren Empfänger in einem Staat, eins oder mehrere identifizierbare Patente in diesem Staat verletzen würden, deren Gültigkeit Ihnen glaubhaft erscheint.

Wenn Sie, als Folge von oder in Verbindung mit einer einzelnen Transaktion oder Vereinbarung, ein betroffenes Werk übertragen oder durch Vermittlung einer Übertragung propagieren, und Sie gewähren einigen Empfängern eine Patentlizenz, die ihnen das Benutzen, Propagieren, Modifizieren und Übertragen einer speziellen Kopie des betroffenen Werks gestatten, dann wird die von Ihnen gewährte Patentlizenz automatisch auf alle Empfänger des betroffenen Werks und darauf basierender Werke ausgedehnt.

Eine Patentlizenz ist „diskriminierend“, wenn sie in ihrem Gültigkeitsbereich die speziell unter dieser Lizenz gewährten Rechte nicht einschließt, wenn sie die Ausübung dieser Rechte verbietet oder wenn sie die Nichtausübung einer oder mehrerer dieser Rechte zur Bedingung hat. Sie dürfen ein betroffenes Werk nicht übertragen, wenn Sie Partner in einem Vertrag mit einer Drittpartei sind, die auf dem Gebiet der Verbreitung von Software geschäftlich tätig ist, gemäß dem Sie dieser Drittpartei Zahlungen leisten, die auf dem Maß Ihrer Aktivität des Übertragens des Werks basieren, und gemäß dem die Drittpartei eine diskriminierende Patentlizenz all denjenigen gewährt, die das Werk von Ihnen erhielten, (a) in Verbindung mit von Ihnen übertragenen Kopien des betroffenen Werks (oder Kopien dieser Kopien) oder (b) hauptsächlich für und in Verbindung mit spezifischen Produkten oder Zusammenstellungen, die das betroffene Werk enthalten, es sei denn, Sie sind in diesen Vertrag vor dem 28. März 2007 eingetreten oder die Patentlizenz wurde vor diesem Datum erteilt.

Nichts in dieser Lizenz soll in einer Weise ausgelegt werden, die irgendeine implizite Lizenz oder sonstige Abwehr gegen Rechtsverletzung ausschließt oder begrenzt, die Ihnen ansonsten gemäß anwendbarem Patentrecht zustünde.
12. Keine Preisgabe der Freiheit Dritter

Sollten Ihnen (durch Gerichtsbeschluß, Vergleich oder anderweitig) Bedingungen auferlegt werden, die den Bedingungen dieser Lizenz widersprechen, so befreien Sie diese Umstände nicht von den Bestimmungen dieser Lizenz. Wenn es Ihnen nicht möglich ist, ein betroffenes Werk unter gleichzeitiger Beachtung der Bedingungen in dieser Lizenz und Ihrer anderweitigen Verpflichtungen zu übertragen, dann dürfen Sie als Folge das Programm überhaupt nicht übertragen. Wenn Sie zum Beispiel Bedingungen akzeptieren, die Sie dazu verpflichten, von denen, denen Sie das Programm übertragen haben, eine Gebühr für die weitere Übertragung einzufordern, dann besteht der einzige Weg, sowohl jene Bedingungen als auch diese Lizenz zu befolgen darin, ganz auf die Übertragung des Programms zu verzichten.
13. Nutzung zusammen mit der GNU Affero General Public License

Ungeachtet anderer Regelungen dieser Lizenz, ist es Ihnen gestattet, ein betroffenes Werk mit einem Werk zu einem einzelnen, kombinierten Werk zu verbinden (linken) oder zu kombinieren, das unter Version 3 der GNU Affero General Public License steht, und das Ergebnis zu übertragen. Die Bedingungen dieser Lizenz bleiben weiterhin auf denjenigen Teil anwendbar, der das betroffene Werk darstellt, aber die speziellen Anforderungen der GNU Affero General Public License, §13, die sich auf Interaktion über ein Computer-Netzwerk beziehen, werden auf die Kombination als solche anwendbar.
14. Überarbeitungen dieser Lizenz

Die Free Software Foundation kann von Zeit zu Zeit überarbeitete und/oder neue Versionen der General Public License veröffentlichen. Solche neuen Versionen werden vom Grundprinzip her der gegenwärtigen entsprechen, können aber im Detail abweichen, um neuen Problemen und Anforderungen gerecht zu werden.

Jede Version dieser Lizenz hat eine eindeutige Versionsnummer. Wenn in einem Programm angegeben wird, daß es dieser Lizenz in einer bestimmten Versionsnummer „oder jeder späteren Version“ (“or any later version”) unterliegt, so haben Sie die Wahl, entweder den Bestimmungen der genannten Version zu folgen oder denen jeder beliebigen späteren Version, die von der Free Software Foundation veröffentlicht wurde. Wenn das Programm keine Versionsnummer angibt, können Sie eine beliebige Version wählen, die je von der Free Software Foundation veröffentlicht wurde.
15. Gewährleistungsausschluß

Es besteht keinerlei Gewährleistung für das Programm, soweit dies gesetzlich zulässig ist. Sofern nicht anderweitig schriftlich bestätigt, stellen die Urheberrechtsinhaber und/oder Dritte das Programm so zur Verfügung, „wie es ist“, ohne irgendeine Gewährleistung, weder ausdrücklich noch implizit, einschließlich – aber nicht begrenzt auf – die implizite Gewährleistung der Marktreife oder der Verwendbarkeit für einen bestimmten Zweck. Das volle Risiko bezüglich Qualität und Leistungsfähigkeit des Programms liegt bei Ihnen. Sollte sich das Programm als fehlerhaft herausstellen, liegen die Kosten für notwendigen Service, Reparatur oder Korrektur bei Ihnen.
16. Haftungsbegrenzung

In keinem Fall, außer wenn durch geltendes Recht gefordert oder schriftlich zugesichert, ist irgendein Urheberrechtsinhaber oder irgendein Dritter, der das Programm wie oben erlaubt modifiziert oder übertragen hat, Ihnen gegenüber für irgendwelche Schäden haftbar, einschließlich jeglicher allgemeiner oder spezieller Schäden, Schäden durch Seiteneffekte (Nebenwirkungen) oder Folgeschäden, die aus der Benutzung des Programms oder der Unbenutzbarkeit des Programms folgen (einschließlich – aber nicht beschränkt auf – Datenverluste, fehlerhafte Verarbeitung von Daten, Verluste, die von Ihnen oder anderen getragen werden müssen, oder dem Unvermögen des Programms, mit irgendeinem anderen Programm zusammenzuarbeiten), selbst wenn ein Urheberrechtsinhaber oder Dritter über die Möglichkeit solcher Schäden unterrichtet worden war.
17. Interpretation von §§ 15 und 16

Sollten der o.a. Gewährleistungsausschluß und die o.a. Haftungsbegrenzung aufgrund ihrer Bedingungen gemäß lokalem Recht unwirksam sein, sollen Bewertungsgerichte dasjenige lokale Recht anwenden, das einer absoluten Aufhebung jeglicher zivilen Haftung in Zusammenhang mit dem Programm am nächsten kommt, es sei denn, dem Programm lag eine entgeltliche Garantieerklärung oder Haftungsübernahme bei.
ENDE DER LIZENZBEDINGUNGEN
Wie Sie diese Bedingungen auf Ihre eigenen, neuen Programme anwenden können

Wenn Sie ein neues Programm entwickeln und wollen, daß es vom größtmöglichen Nutzen für die Allgemeinheit ist, dann erreichen Sie das am besten, indem Sie es zu freier Software machen, die jeder unter diesen Bestimmungen weiterverbreiten und verändern kann.

Um dies zu erreichen, fügen Sie die folgenden Vermerke zu Ihrem Programm hinzu. Am sichersten ist es, sie an den Anfang einer jeden Quelldatei zu stellen, um den Gewährleistungsausschluß möglichst deutlich darzustellen; zumindest aber sollte jede Datei die „Copyright“-Zeile besitzen sowie einen kurzen Hinweis darauf, wo die vollständigen Vermerke zu finden sind.

    [eine Zeile mit dem Programmnamen und einer kurzen Beschreibung]
    Copyright (C) [Jahr] [Name des Autors]

    This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

    This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

    You should have received a copy of the GNU General Public License along with

this program; if not, see <http://www.gnu.org/licenses/>.

Auf Deutsch:

    [eine Zeile mit dem Programmnamen und einer kurzen Beschreibung]
    Copyright (C) [Jahr] [Name des Autors]

    Dieses Programm ist freie Software. Sie können es unter den Bedingungen der GNU General Public License, wie von der Free Software Foundation veröffentlicht, weitergeben und/oder modifizieren, entweder gemäß Version 3 der Lizenz oder (nach Ihrer Option) jeder späteren Version.

    Die Veröffentlichung dieses Programms erfolgt in der Hoffnung, daß es Ihnen von Nutzen sein wird, aber OHNE IRGENDEINE GARANTIE, sogar ohne die implizite Garantie der MARKTREIFE oder der VERWENDBARKEIT FÜR EINEN BESTIMMTEN ZWECK. Details finden Sie in der GNU General Public License.

    Sie sollten ein Exemplar der GNU General Public License zusammen mit diesem Programm erhalten haben. Falls nicht, siehe <http://www.gnu.org/licenses/>.

Fügen Sie auch einen kurzen Hinweis hinzu, wie Sie elektronisch und per Brief erreichbar sind.

Wenn Ihr Programm interaktive Befehle in einem Terminal entgegennimmt, sorgen Sie dafür, daß es nach dem Start einen kurzen Vermerk ausgibt:

    [Programm] Copyright (C) [Jahr] [Name des Autors]
    This program comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
    This is free software, and you are welcome to redistribute it under certain conditions; type ‘show c’ for details.

Auf Deutsch:

    [Programm] Copyright (C) [Jahr] [Name des Autors] Für dieses Programm besteht KEINERLEI GARANTIE; geben Sie "show w" für Details ein.
    Dies ist freie Software, die Sie unter bestimmten Bedingungen weitergeben dürfen; geben Sie "show c" für Details ein.

Die hypothetischen Kommandos „show w“ und „show c“ sollten die entsprechenden Teile der GNU-GPL anzeigen. Natürlich können die von Ihnen verwendeten Kommandos auch anders lauten; für ein Programm mit graphischer Benutzeroberfläche werden Sie sicherlich eine „About-Box“ verwenden.

Soweit vorhanden, sollten Sie auch Ihren Arbeitgeber (wenn Sie als Programmierer arbeiten) oder Ihre Schule einen Urheberrechteverzicht für das Programm unterschreiben lassen. Für weitere Informationen darüber und wie Sie die GNU GPL anwenden und befolgen, siehe http://www.gnu.org/licenses/.

Diese General Public License gestattet nicht die Einbindung Ihres Programms in proprietäre Programme. Wenn Ihr Programm eine Funktionsbibliothek ist, dann kann es sinnvoller sein, das Linken proprietärer Programme mit dieser Bibliothek zu gestatten. Wenn dies Ihre Absicht ist, sollten Sie die GNU Lesser General Public License anstelle dieser Lizenz verwenden. Lesen Sie aber bitte vorher http://www.gnu.org/philosophy/why-not-lgpl.html.

* * *

Copyright-Notiz des englischsprachigen Orignals:

Apache Commons

Teile von BASIC! nutzen Apache Commons.