

QBASIC

Diese Anleitung beschreibt das BASIC-Interface zwischen QD und QLiberator. Um es nutzen zu können benötigen Sie also auf jeden Fall QD (Version 5 oder neuer) und QLiberator (vorzugsweise V3.35 oder neuer). In der Beschreibung dieser Anleitung wird vorausgesetzt, daß Sie die Anleitungen von QD und QLiberator gelesen haben - trotzdem beschreibt sie hier die Vorgänge noch einmal detailliert. Wo möglich wird auf die jeweilige Haupt-Anleitung verwiesen.

Das Prinzip: QBASIC ist eine residente Erweiterung, die als solche dann natürlich mit LRESPR geladen werden muß. Vorher muß die Erweiterung jedoch konfiguriert werden damit die Einstellungen Ihrem System entsprechend stimmen. Wenn QBASIC geladen ist installiert es ein Thing für QD (siehe auch in der QD-Anleitung). Wenn ein BASIC-Programm eingegeben ist, kann dieses Thing mit F10 aufgerufen werden und kann dann den Parser aktivieren. Der Parser überprüft das Programm auf Fehler - ist es Syntax-Fehler-frei wird eine Datei angelegt die das Format hat das QLiberator bearbeiten kann (also eine Art Workfile, wie sie auch mit QSAVE bzw. dem LIBERATE-Befehl angelegt wird). Das Programm wird dann compiliert und gestartet. QD bleibt da - es kann weiter editiert werden während das Programm läuft - es können auch weitere Programme aus dem Source-Text generiert und gestartet werden.

Nun zur Vorgehensweise: zuerst muß die Datei **QBASIC_rext** konfiguriert werden. Starten Sie dazu das **CONFIG** bzw. **MenuConfig** Programm und laden Sie die Datei **QBASIC_rext**. Sie finden drei einstellbare Menüpunkte vor sich die alle mit dem QLiberator zu tun haben:

QLiberator Thingname: QLIB. Wenn Sie QLiberator als Thing geladen haben (oder laden werden) geben Sie hier bitte den Thing-Namen an. Es gibt viele verschiedene Möglichkeiten QLiberator als Thing zu installieren - kennen Sie keine davon haben Sie wahrscheinlich nicht die Möglichkeit dies zu tun. Es ist auch nicht schlimm, da QBASIC automatisch nach einer QLiberator-Datei sucht wenn es kein QLiberator-Thing findet.

QLiberator Dateiname: flp1_QLIB_obj. Hier geben Sie den Dateinamen an den der QLiberator in Ihrem System hat. Wenn Sie die Original-QLiberator-Diskette benutzen ist es flp1_QLIB_obj. Wenn Sie den QLiberator auf Festplatte gespeichert haben geben Sie den vollständigen Dateinamen an.

QLiberator Optionen: leer. Hier können Sie die Optionen definieren die beim Compiler-Aufruf an den QLiberator übergeben werden sollen (beispielsweise -NOWINDS). Schauen Sie bitte in die QLiberator-Anleitung für die Erklärung der verschiedenen Compiler-Optionen.

Das war es schon, was das Konfigurieren anging. Die so richtig voreingestellte Datei kann resident geladen werden.

Starten von QD mit BASIC-Möglichkeit: Wenn Sie QD aufrufen möchten und QBASIC darin benutzen wollen, dann müssen Sie das QD beim Start mitteilen. Dies geht relativ einfach, nämlich über den Befehlsstring (genauere Beschreibung des Befehlsstrings in der QD-Anleitung). Beispiele:

EXEP QD;"\T QBASIC" falls QD resident geladen ist oder

EX flp1 QD;"\T QBASIC" startet QD von Diskette.

Dies startet QD und benutzt das QBASIC Thing. Wenn Sie eine vollständige BASIC-Umgebung haben wollen, beispielsweise auch mit dem HyperHelp BASIC System aus QD, dann nehmen Sie

EXEP QD;"\TQBASIC \Hflp1 basic hilfe \E bas \Dflp1 BASIC "

Das \T definiert das Thing QBASIC, \H definiert wo das Hilfssystem zu finden ist (wenn Sie es auf Harddisk haben geben Sie hier das jeweilige Unterverzeichnis an), \E stellt die Datei-Endung _BAS ein (Sie möchten ja BASIC-Programme laden und speichern) und \D gibt den Verzeichnis-Vorschlag an, also flp1_BASIC_.

Wie auch immer Sie QD gestartet haben, wenn es läuft müssen Sie hinter dem F10-Menüpunkt den Text QBASIC sehen. Falls nicht haben Sie beim Aufruf etwas falsch gemacht.

Innerhalb von QD können Sie beliebig BASIC-Programme laden, speichern, editieren oder neu eingeben. Als Besonderheit ist es auch möglich, zeilennummernlose Programme einzugeben!

Wenn Sie das BASIC-Programm compilieren möchten das sich gerade im QD befindet, dann drücken Sie **F10**. Passiert nichts, haben Sie das QBASIC nicht oder nicht richtig installiert. Im positiven Fall erscheint nun ein Menü, aus dem Sie auswählen können ob Sie die vordefinierten Compiler-Optionen ändern möchten (z.B. -NOWINDS Option eingeben etc. - siehe QLiberator Anleitung) oder das Programm nur compilieren möchten oder compilieren und das Objekt-Programm starten.

Probieren wir es einfach mal. Geben Sie ein:

```
FOR x=32 TO 191
  PRINT x,CHR$(x)
END FOR x
INPUT ende$
```

Drücken Sie nun **F10 Starten** und los geht's. Ein QBASIC-Fenster zieht sich links oben auf (und nur da wird es sich aufziehen, da QLiberator auch immer darüber heraus kommt, leider) und nach kurzer Zeit erscheint QLiberator, der das Programm compiliert. Da er im oben angegebenen Programm keinen Fehler finden wird, sofern Sie es richtig eingegeben haben, wird dann sofort das Programm gestartet, die Zeichen ausgegeben und auf einmal ENTER oder RETURN gewartet. Das Programm verschwindet nun.

Provozieren wir nun einen Fehler. Fügen Sie an:

```
q==1
```

bringen Sie den Cursor dann irgendwo ins Bild um den Effekt zu sehen und drücken Sie wieder **F10 Starten**. Sofort wird der Cursor über das zweite '=' gebracht, da der Parser meint dort einen Syntax-Fehler gefunden zu haben (und damit ja auch Recht hat).

Oder probieren Sie mal:

```
PRINT HEX$(12345,16
```

und versuchen Sie wieder **F10 Starten**. Der Cursor wird ans Zeilenende gebracht da der Parser eine geschlossene Klammer vermißt, und die gehört nun mal am wahrscheinlichsten ans Zeilen-Ende. Sie hätte natürlich nach dem Parameter 12345 stehen können, aber die HEX\$-Funktionen braucht in diesem Fall ja zwei Parameter und der Parser kann nur raten. Anders hingegen bei:

```
PRINT LEN('123':PRINT LEN('4567')
```

Hier wird der Cursor hinter den String '123' gesetzt, auf den Doppelpunkt, da hier an Stelle des Doppelpunktes erst mal eine geschlossene Klammer hingehört, dann kann man weiter sehen. Sie sehen, der Parser versucht sein Bestes.

Dinge, die der Parser nicht mag, sind: Zeilen die länger als die maximale Zeilenlänge in QD sind (also mit dem → Symbol beginnen). Dies geht nicht! Wenn Sie längere Zeilen als voreingestellt benötigen (normalerweise 160 Zeichen), dann konfigurieren Sie QD auf mehr Zeichen pro Zeile, schon geht's.

Aber weiter mit der Funktionsweise von QBASIC: Starten Sie noch einmal obiges (fehlerfreies) Programm und drücken Sie keine Taste wenn die Zeichen ausgegeben wurden. Schauen Sie mal in die Job-Liste. Sie sehen hier einen Job namens "RAM1_". Da Sie dem Programm noch keinen Namen gegeben haben bekommt es irgend einen. Dem ist einfach abzuhelfen: speichern Sie das Programm einmal ehe Sie es compilieren (sollten Sie sowieso machen, es könnte ja crashen und dann ist alles Eingeebene weg!), z.B. als **RAM1_TEST_BAS**. Raten Sie mal, wie das Ergebnis heißt (ist ja nicht schwierig es herauszufinden - **F10 Starten**). Na, es heißt nur "TEST" - so sollte es sein. _BAS ist ja nur eine Datei-Namen-Endung damit Sie BASIC-Programme einfacher auffinden können, und das Laufwerk interessiert im Job-Namen auch nicht.

Schauen Sie mal in RAM1_. Es gibt dort mindestens drei Dateien:

TEST_BAS

haben Sie ja gerade eben gespeichert.

TEST_OBJ

das fertige Objekt-Programm.

TEST_ERR

enthält Fehler und Warnungen, diesmal jedoch nicht.

Die **OBJ**-Datei ist ausführbar, Sie können Sie nun also beliebig oft starten (EX, QX, QPAC 2's Dateien oder Cueshell, beispielsweise) ohne über QD gehen zu müssen.

Die **ERR**-Datei enthält keinen Fehler, sonst hätte der Parser es Ihnen ermöglicht einen kurzen Blick darauf zu werfen. Aber dies ist schnell erledigt: Fügen Sie an das BASIC-Programm folgende Zeile an und versuchen Sie es zu starten:

```
fred 500
```

mit der Annahme, daß Sie keine Prozedur namens fred definiert haben, meldet QLiberator "ambiguous name". Vorher jedoch beendet er das Compilieren und läßt Sie erst mal die SPACE-Taste drücken. Reichlich überflüssig, aber das kann man beseitigen. Auf der Diskette befindet sich ein BASIC-Programm mit dem man QLIB_obj patchen kann, aber dazu gleich mehr. Haben Sie nun SPACE gedrückt erscheint ein Fenster in dem die Fehlermeldungen erscheinen. Sie können durchscrollen bis Sie gefunden haben was alles falsch ist, in unserem Fall gibt's jedoch nicht besonders viel zu scrollen. Dieses Fenster erscheint bei jedem Fehler, man kann ein fehlerhaftes Programm, das nicht erfolgreich compiliert wurde ja auch nicht risikolos starten und sollte dies auch nicht versuchen.

Bleiben noch die Warnungen: schreiben Sie am besten ein Programm das aus folgenden Zeilen besteht:

```
a=3
```

```
OPEN #a,CON
```

```
CLS#a
```

```
INPUT #a,a$
```

und versuchen Sie, es zu starten. QLiberator wartet wieder auf einen SPACE-Tastendruck sofern Sie ihn nicht schon gepatcht haben weil es so stört, dann erscheint im QBASIC-Fenster eine Auswahlbox: Editor bringt Sie zurück in den Editor, Startversuch macht genau das, was es aussagt, Sie müssen jedoch mit Compiler-Runtime-Fehlern rechnen (aber all dies ist ja sehr ausführlich in der QLiberator-Anleitung beschrieben) oder Anschauen - genau wie beim Fehlerfall wird die _ERR-Datei angezeigt.

QLiberator patchen

Auf der Diskette finden Sie auch die Datei **PATCHQLIB_bas**. Hiermit können Sie die lästige SPACE-Nachfrage im Fehler- oder Warnfall beseitigen, außerdem werden die Farben so gepatcht, daß QLIB, wie wir finden, besser optisch in die Gesamt-Umgebung paßt (grün vorausgesetzt). Lassen Sie das Programm einfach laufen, aber verändern Sie bitte nur eine Arbeitskopie, NICHT das Original. Patchen Sie auch nur Version 3.35 (nicht wundern, auch V3.36 hat ein QLIB_obj mit V3.35 - die Runtimes sind 3.36).