VBDis3 - *DER* Discompiler für Visual Basic 3.0

Dieser Discompiler erzeugt aus compilierten Visual Basic (VB) Programmen die kompletten Quelldateien, einschließlich der Forms. Er soll in dieser Version nicht dazu dienen, fremde Programme zu knacken, sondern demonstrieren, wieviele Teile des Sourcecodes vom Compiler in die ausführbaren Programme übernommen wurden. Ich halte dies für ein bedenkliches Zeichen, für welches Programm sind denn diese Informationen bestimmt? Das Laufzeitsystem benötigt sie nicht, und doch wurden sie teilweise absichtlich in das Programm übernommen.

Die Installation des Discompilers ist einfach:
Kopieren Sie das Archiv in ein neues Verzeichnis (z.B. VBDIS) und packen Sie es aus. Wenn Sie dies bereits getan haben, finden Sie neben dem Programm **VBDIS3.EXE** noch einige Datenfiles, die für die Discompilierung benötigt werden. Eine Installation ist nicht notwendig, der Discompiler benötigt keine Einträge in WIN.INI usw.

Die Bedienung des Discompilers ist ebenfalls einfach:
- Programm starten, aus dem Menü 'Öffnen' wählen und das gewünschte Programm angeben.
- Das Verzeichnis wählen, in dem die Quellen erzeugt werden sollen. Mit 'Neu' kann ein neues Verzeichnis mit dem darüber angegebenen Namen erstellt werden. Der gewähte Ordner muß vor dem Verlassen dieses Dialogs geöffnet dargestellt sein!
- Nun legt der Discompiler los und erzeugt die Quellfiles, sofern es sich um ein Programm von VB 3.0 handelt. Sollten Sie versuchen, ein anderes Programm, den Discompiler selbst oder ein anderes geschütztes Programm zu discompilieren, erscheint eine Fehlermeldung.
- Im ersten Durchlauf erzeugt der Discompiler mehrere BAS-Files im Textformat sowie die FRM-Files, diese allerdings im Binärformat. Sobald er fertig ist, rufen Sie VB mit dem erzeugten Makefile auf und speichern alle Forms ebenfalls im Textformat. Dieser Schritt ist leider notwendig, da die Erzeugung der Forms im Textformat ohne Mithilfe der Custom-Controls (VBX) oft nicht möglich ist. Speichern Sie dann das Projekt ab und verlassen Sie den Interpreter.
- Anschließend wählen Sie im Menü des Discompilers 'Forms zusammenfügen', und der Code wird an die Forms angehängt. (Geht natürlich auch mit einem Editor, aber nicht so schnell).
- Fertig - das Programm kann nun im Interpreter gestartet werden.

Während der Discompilierung können Fehler gemeldet werden, die (hoffentlich) abgefangen werden und das Programm nicht abstürzen lassen. Bitte beachten Sie die Kommentare hierzu, die zugrundeliegenden Tabellen wurden ohne Disassemblierung des Laufzeitsystems erstellt, so daß sie nicht unbedingt komplett sind. Auch einige Konstruktionen in den Programmen können zu Fehlern führen, sofern sie bislang noch nicht aufgetreten sind und daher nicht bekannt ist, wie sie zu behandeln sind.

Die Beschreibungen von Custom-Controls liegen in Datenfiles mit der Extension '300' (für VB 3.0). Sollte das Programm neben den mitgelieferten Controls weitere VBX-Dateien verwenden, kann der Discompiler diese natürlich nicht richtig interpretieren. Hier hilft das Programm **VBCtrl**, das VBX-Dateien analysiert und für die Verwendung im Discompiler abspeichert. Wenn der Discompiler auf ein unbekanntes Control stößt, können Sie **VBCtrl** starten, die Beschreibung des Controls abspeichern und anschließend weiter discompilieren.

Beim Starten des Programms kann der Interpreter einige Fehler finden, hauptsächlich fehlende oder falsche Variablentypen, sowie falsche Argumente beim Aufruf von Funktionen. Diese Angaben müssen Sie selbst ermitteln und korrigieren, die Vergabe von Namen und Datentypen für die Variablen und Funktionen ist nur in der Profi-Version möglich. Beim Vergleich mit Ihrem Quellcode entdecken Sie möglicherweise fehlende Variablen und Konstanten, sowie seltsame Parameterlisten von Unterprogrammen (p,p,p); diese Symptome stammen alle von Variablen bzw. Unterprogrammen und Parametern, die im Programm garnicht verwendet werden, so daß der Discompiler die Datentypen nicht ermitteln kann.

Beachten Sie im Quellcode die Namen der Forms und Controls, die der Compiler in das ausführbare Programm hineingepackt hat, und die exakte Einrückung aller Zeilen, die ebenfalls mit viel Aufwand aus dem Quellcode übernommen wurde. Bei jeder Variablen wird peinlich genau vermerkt, ob sie im Quellcode mit oder ohne Typ-Kennzeichen angegeben wurde. Sie können das mit eigenen Programmen ganz leicht austesten und die erzeugten Quellen mit Ihrem Code vergleichen.

VBCtrl - *DAS* Custom Control Tool

VBCtrl analysiert EXE- und VBX-Dateien und extrahiert die Properties und Events der darin gespeicherten Controls. Die Bedienung erfordert ein paar Kenntnisse, die hier im Schnelldurchgang vermittelt werden sollen.

Starten Sie **VBCtrl** und öffnen Sie die gewünschte Datei. Alle Dateien vom Typ VBX werden automatisch näher analysiert und es erscheint ein Fenster mit mehreren Listen. In der linken Liste sind alle möglichen Entrypoints der Datei aufgelistet. Ihre Aufgabe besteht nun darin, alle Prozeduren zu finden, die zu den Custom Controls gehören.

In den meisten VBX sind die Control-Prozeduren am Namen xxxCTRLPROC zu erkennen (brav vom CDK abgeschrieben). Wählen Sie nun den ersten dieser Namen an. Daraufhin erscheint im mittleren Fenster eine Liste aller Verweise auf diese Prozedur, von denen eine die gesuchte Control-Tabelle sein sollte. Wenn Sie keinen entsprechenden Namen finden, oder wenn die vermeintlichen Tabellen nicht richtig interpretiert werden können, probieren Sie der Reihe nach alle (unbenannten) Entrypoints, normalerweise werden Sie auch dann irgendwann einmal fündig.

Wählen Sie nun aus dem mittleren Fenster eine Adresse aus, im rechten Fenster wird dann die vermutete Tabelle dargestellt. Bei Fehlermeldungen ist die Wahrscheinlichkeit sehr gering, daß Sie eine solche Tabelle gefunden haben. Achten Sie auf den Klassennamen (CN) und Default-Control-Namen (DN), hier sollten lesbare Texte erscheinen. Am ersten Eintrag können Sie noch die VB-Version unterscheiden (100=VB1.0, 200=VB2.0, 300=VB3.0), hier sollte unbedingt einer dieser Werte stehen. Einzelheiten zu den angezeigten Werten lesen Sie bitte im CDK nach, jede Zeile entspricht einem Eintrag in der Control-Model Struktur, die Buchstaben am Anfang jeder Zeile sind Kürzel für die entsprechenden Elemente in der Struktur. Bei mehreren Tabellen mit denselben Control-Namen wählen Sie diejenige mit der höchsten Versionsnummer.

Mit dem Button oberhalb der rechten Liste übernehmen Sie alle Informationen, die zu diesem Steuerelement gehören, in den Katalog, der vom Discompiler für die Behandlung des Steuerelements benötigt werden. Dieser Katalog wird in einem weiteren Fenster angezeigt.
Die Anzeige der Properties und Events ermöglicht kleinere Korrekturen, die Sie selbst bei Bedarf durchführen können. Speziell in den Standard-Controls (die in VBRUN300 implementiert sind), fehlen die Namen der Parameter zu den Events, einige Datentypen sind nicht dokumentiert und müssen von Hand nachgetragen werden. Hierzu dient das Edit-Feld über der Event-Liste, in das Sie die gewünschten Parameter-Strings eintragen können. Dies erfolgt am einfachsten über den Hilfe-Button, mit dem die Information zu dem aktuellen Event angezeigt wird. Dort können Sie die Parameterliste ausschneiden und dann in das Edit-Feld einfügen.
Die Standard-Properties sind alles andere als 'Standard', die Datentypen sind nirgends dokumentiert, und einige Namen fehlen. Dies sollte jedoch nicht weiters stören, da diese Fälle im Discompiler automatisch abgefangen werden bzw. die Properties ohne Namen in einem Programm ja garnicht vorkommen können. Auch die Standard-Collections werden im Discompiler automatisch berücksichtigt, und für Custom-Controls gibt es keine Möglichkeit, Collections zu definieren oder zu verwenden. Sie sollten jedoch die mitgelieferten Beschreibungen der Standard-Controls (VBRUN300.300) möglichst nicht verändern, da dort Informationen gespeichert sein können, deren Erzeugung mit **VBCtrl** nicht garantiert werden kann.

Wenn Sie so alle Controls einer VBX-Datei gefunden und katalogisiert haben, speichern Sie die Definitionen ab. Dabei wird eine Datei mit dem Namen der VBX-Datei und der Erweiterung '.300' erzeugt, mit welcher der Discompiler alle Stellen richtig verarbeiten, an denen die Controls in einem Programm verwendet werden.

Gemeinerweise habe ich mich entschlossen, **VBCtrl** zunächst so **einzuschränken**, daß die angezeigten Angaben **nicht** abgespeichert werden. Dies ermöglicht zwar das Austesten des Discompilers, aber fast nur mit eigenen Programmen, die keine speziellen Custom-Controls enthalten. Ich betrachte dies auch als einen Schutz aller bislang erschienenen Programme gegen Discompilierung, aber wer seine Programme in Zukunft nicht gegen Discompilierung schützt, gibt seine Quellen jedem Interessenten in die Hand, der über einen Discompiler für Visual Basic verfügt!

In der Lite-Version wird die Abspeicherung in Verbindung mit der Optimierung der Programme möglich sein, ich bitte alle Anwender noch um etwas Geduld.

Hinweise

Die aktuelle Version des Discompilers ist 3.38 vom Juli1995, alle älteren Versionen können sie getrost vergessen. Hierbei handelt es sich noch um eine Beta-Version, die nächste offizielle Version wird 3.40 sein. Kleinere Anpassungen können durch Auswechseln der Tabellen erfolgen, ohne daß das Programm ausgetauscht werden muß. Dies kommt speziell in Frage, wenn der Discompiler unbekannte Tokens meldet. In diesem Fall sollten Sie zunächst prüfen, ob Sie die neuesten Tabellen verwenden, und ggf. das Programm, das zu dieser Fehlermeldung geführt hat, an den Autor einsenden, damit die Tabellen auf den neuesten Stand gebracht werden können.

Neben der Demo-Version gibt es eine **Lite-Version** des Discompilers, die gegenüber der Demo-Version folgende Vorteile bietet:
- Kein Abbruch bei unbekannten Custom-Controls
- Keine Verzögerung des Programmablaufs
- 1 Jahr lang kostenlose Updates

Diese Version können Sie mit dem beiliegenden Registrierungs-Formular bestellen.

Immer noch in Arbeit ist eine **Profi-Version**, die u.a. folgende Möglichkeiten bieten soll:
- Bearbeitung sehr großer Programme
- Manuelle Vorgabe von Namen und Datentypen für Variable und Unterprogramme
- Vergleich des erzeugten Codes mit vorhandenen Quelltexten
- Behandlung aller Custom-Controls
- Hinweise auf Optimierungsmöglichkeiten im Programm
- Optimierung der compilierten Programme
- Ausgabe des Programms in anderen Programmiersprachen (C++, Pascal)

Einige dieser Optionen können noch in die Lite-Version übernommen werden, die Einzelheiten müssen jedoch noch festgelegt werden. Die Profi-Version soll professionell vertrieben werden, und die Verhandlungen sind noch im Gange. Die Optimierung eigener Programme soll auch in der Lite-Version möglich sein, der Umfang ist jedoch noch nicht geklärt. Da diese Funktion eigentlich zu den VB-Tools gehört und nicht zum Discompiler, wird in Zukunft wohl nur noch ein gemeinsames Paket mit den Tools und dem Discompiler erscheinen, die meisten Anwender haben sich ja gleich für beides registrieren lassen.

Ausblick

Der Discompiler wird bei der Optimierung Ihrer Programme noch eine wichtige Rolle spielen. Die Entwicklungsversion gestattet z.B. den direkten Vergleich des Sourcecodes mit dem compilierten Code, so daß unbekannte Tokens leicht zugeordnet werden können. Dies kann jeder Entwickler selbst durchführen, da diese Informationen alle in Datenfiles gespeichert sind, Änderungen im Programm sind nicht zu erwarten. Weiterhin können die globalen und lokalen Speicherbereiche aufgelistet werden, die Datentypen und Variablennamen können von Hand vorgegeben oder aus den Quellen übernommen werden. Dadurch wird der discompilierte Code vollständiger und kann leichter mit dem Originaltext verglichen werden. Auf Knopfdruck werden sogar alle Namen aus den Quelltexten in die Anzeige der EXE-Datei übernommen.

Die Anzeige der EXE-Datei listet alle Tokens mit einer kurzen Beschreibung auf. Dort finden Sie dann so sinnlose Angaben wie den Typ eines Feldes, wobei peinlich genau die Spalte in der Zeile angegeben ist, in der diese Angabe im Quelltext stand (!), und sogar der Verweis auf den Typ-Namen ist dort gespeichert, was leider wegen der fehlenden Namenstabelle keinerlei brauchbare Informationen liefern kann (Felder sind sowieso im Variablenbereich genau beschrieben, dieses Token kann mit allen zugehörigen Informationen ersatzlos gestrichen werden).
Weitere Ansatzpunkte bei der Optimierung sind unbenutzte Variablen und Funktionen sowie überflüssige Typ-Konvertierungen, die ebenfalls ersatzlos gestrichen werden können. Auf einige Optimierungen haben Sie durch Auswahl der geeigneten Funktionen und Datentypen einen Einfluß, andere Änderungen können nur per Programm vorgenommen werden, was in der nächsten Ausbaustufe geschehen soll. Die ersten Ansätze finden sich im Projekt-Manager, der u.a. überflüssige Namen aus den compilierten Programmen entfernt und sie damit auch schon etwas gegen Discompiler schützt. Alleine diese Funktion sollte jeder Programmierer zum Anlaß nehmen, sich auch mit meinen übrigen VB-Tools zu beschäftigen.

Und damit wären wir beim letzten Punkt, nämlich der Weiterentwicklung der VB-Tools. Es wurde bereits eine Menge Vorarbeit geleistet, doch die Umsetzung der vorhandenen Entwicklungsversionen in allgemein verwendbare Programme erfordert noch einigen Aufwand. Ich betrachte die Anzahl der Registrierungen als ein Maß für das Interesse an der Weiterentwicklung der VB-Tools und bitte daher jeden ernsthaften Interessenten, mit Fehlermeldungen, Vorschlägen und

nicht zuletzt mit seiner Registriergebühr dafür zu sorgen, daß die Weiterentwicklung zu immer besseren Entwicklungswerkzeugen auch tatsächlich stattfinden wird.

Mit dem Erscheinen von Windows 95 und Visual Basic 4.0 stehen neue Aufgaben ins Haus, so daß die Entwicklung der Tools zu VB 3.0 bis dahin abgeschlossen sein soll. Dann steht zunächst die Portierung der vorhandenen Programme auf die neue Version ins Haus, und die Tools müssen an die neuen Gegebenheiten angepaßt werden. Einige Funktionen sind bereits in VB 4.0 enthalten, andere müssen in Add-In's zu VB 4.0 konvertiert werden.

Und nun viel Spaß mit dem VB-Discompiler wünscht

DoDi
(MausNet: Hans-Peter Diettrich @ S)