Comal – eine Einführung

Ist Ihnen Basic zu wenig leistungsfähig, Pascal zu ermüdend, Assembler zu primitiv und Forth zu merkwürdig ? Dann wird es Zeit für Comal.

Teil 1

Die Programmiersprache Comal ist keine ganz neue Sprache, wenngleich sie auch – ähnlich wie Forth – erst seit relativ kurzer Zeit in Europa kursiert. Comal (COMmon Algorithmic Language) wurde bereits 1973 von Borge R. Christensen und Benedict Loefstedt in Dänemark entwickelt, fand aber, wie viele andere Programmiersprachen, zunächst kein großes Echo. Ähnlich wie bei Forth bildeten sich aber bald in vielen Ländern nationale User-Groups, die sich für die Verbreitung und kontinuierliche Verbesserung der Sprache einsetzten.

Das im 64'er Magazin, Ausgabe 8/84, besprochene Comal ist die Version 0.14, das auch als Grundlage für diesen Einführungskurs dient. Comal 2.0 ist um einiges schneller und ist ein 52 KByte langes Programm. Dabei sind mehr als 30 KByte frei für eigene Programme. Das Modul selbst enthält 64 KByte ROM. In Dänemark wird es für umgerechnet 600 Mark angeboten und ist dort fast an allen Schulen im Einsatz. Ältere Comal-Versionen als 0.14 (zum Beispiel 0.12) verfügen unter Umständen über eine Reihe von Befehlen nicht. Umgekehrt ist die Version 0.14 gegenüber der kommerziellen Version 2.0 mit einigen Nachteilen behaftet, zu denen vor allem der mit exakt 9902 Bytes nicht gerade üppig bemessene Speicherplatz gehört. Die Comal-Versionen 0.xx sind sogenannte »Public Domain Software«, das heißt sie dürfen kopiert und weitergegeben, allerdings nicht kommerziell vertrieben werden. Der Sinn dieser auf den ersten Blick verblüffenden Freizügigkeit liegt in der gewünschten möglichst umfassenden Verbreitung der Sprache.

In letzter Zeit gewinnt Comal zunehmende Bedeutung als Alternative zur vergleichsweise primitiven Basic-Programmierung. Der Benutzer von Comal hat dabei gegenüber der Verwendung anderer Programmiersprachen den entscheidenden Vorteil, daß die Sprache stark an Basic angelehnt ist, daß aber die vielen Schwachpunkte von Basic überwunden wurden. Hinsichtlich des Konzeptes der strukturierten Programmierung wurden viele Anleihen bei Pascal gemacht, ohne jedoch dessen oft unnötig komplizierte und langatmige Sprachstruktur zu kopieren.

Doch damit noch nicht genug. Jetzt wird es für den Besitzer eines C 64 erst richtig interessant (für den VC 20 ist Comal leider nicht erhältlich). Comal hat nämlich noch bei einer dritten Programmiersprache Anleihen gemacht, und zwar bei Logo. Damit stehen eine ganze Anzahl von sehr stark an Logo angelehnten Befehlen für hochauflösende Grafik zur Verfügung. Der große Vorteil dieser Befehle ist ihre einfache und übersichtliche Struktur. Es sind kaum Koordinatenangaben nötig, sondern es wird mit einer sogenannten »Turtle« gearbeitet.

Turtle ist die englische Bezeichnung für »Schildkröte«. Dieser etwas ungewöhnliche Name stammt aus Logo, das ursprünglich als grafische Programmiersprache für Kinder entwickelt wurde. Die »Schildkröte« ist dabei ein Grafik-Cursor (bei Comal durch ein Dreieck dargestellt), der auf den gerade aktuellen Punkt im Grafik-Bildschirm zeigt. Die Turtle kann nun mit verschiedenen einfachen Befehlen bewegt werden. FORWARD n bewegt die Turtle um n Grafikpunkte vorwärts, BACK n bewegt sie rückwärts. LEFT w und RIGHT w drehen die Turtle um den Winkel w (in Grad) nach links oder rechts. HOME setzt die Turtle wieder auf ihre Ausgangsposition in der Bildschirmmitte.

Es existieren natürlich noch eine ganze Reihe weiterer Grafikbefehle und sogar etliche Befehle zur Spritesteuerung. Das kleine Demo-Programm (siehe Listing) zeichnet einige Quadrate mit zunehmender Seitenlänge auf den Bildschirm. An diesem Programm sind schon einige grundsätzliche Eigenschaften von Comal erkennbar, zum Beispiel die Programmstrukturierung und das Arbeiten mit Prozeduren. Wenn Sie Comal bereits zur Verfügung haben, dann geben Sie doch dieses kleine Programm einfach einmal ein und lassen sich überraschen.

Wir wollen uns an dieser Stelle jedoch noch nicht weiter mit Einzelhei-

ten wie Prozeduren oder Funktionen beschäftigen, sondern uns dieses sehr umfassende Thema für später aufheben. Auch Comal-Grafik und Sprites werden wir ausführlich in einer der nächsten Folgen behandeln. Wir wollen stattdessen ganz am Anfang beginnen und uns etwas genauer damit beschäftigen, wie man denn nun Comal dazu bekommt, die grundlegenden Dinge wie Programm editieren, laden, speichern und drucken für uns zu erledigen.

Das Arbeiten mit Comal

Kommen Sie mit Basic zurecht? Dann wird Ihnen auch das Arbeiten mit Comal keine Schwierigkeiten bereiten. Genau wie Basic ist auch Comal eine interaktive Sprache. Die meisten Befehle lassen sich auch im Direktmodus ausführen. Soll jedoch eine Programmzeile gespeichert werden, so setzt man einfach — wie von Basic bekannt — eine Zeilennummer davor.

Doch Vorsicht! Einige Unterschiede zu Basic gibt es schon beim Editieren eines Programms. Zum Beispiel dürfen in einer Comal-Zeile nicht mehrere Befehle stehen (der Doppelpunkt fungiert in Comal nicht als Trennzeichen zwischen zwei Befehlen, sondern hat verschiedene andere Funktionen). Auch ist es zum Löschen einer Programmzeile nicht ausreichend, nur die Zeilennummer einzugeben. Dafür gibt es den »DEL«-Befehl, mit dem man nicht nur einzelne Zeilen, sondern auch ganze Zeilenbereiche löschen kann. Die Syntax ist die gleiche wie bei »LIST«. »DEL 50-150« löscht also beispielsweise die Zeilen 50 bis 150. Der »LIST«-Befehl ist in Comal übrigens sehr komfortabel. Das Listen eines Programms läßt sich nämlich durch Drücken der Space-Taste anhalten. Ein zweiter Tastendruck, und das Listen wird fortgesetzt.

Zu beachten ist auch, daß Zeilennummern in Comal maximal vierstellig sein dürfen, sonst wird ein Syntax-Fehler angezeigt. Auch die Null ist als Zeilennummer nicht erlaubt.

Bei der Programmeingabe wird man sehr schnell eine wesentliche Eigenschaft von Comal kennen- und schätzenlernen, nämlich den sofortigen Syntax-Check. Alle fehlerhaften Programmzeilen werden bereits bei der Eingabe mit einer entsprechenden Fehlermeldung zurückgewiesen. Der Cursor blinkt dabei genau an der Stelle, an welcher der Fehler aufgetreten ist.

Interpreter oder Compiler?

Comal führt die meisten Befehle im Direktmodus aus, die Programme werden einfach mit »RUN« gestartet — wie bei einem typischen Interpreter. Andererseits müssen Strings dimensioniert und Variablen vor dem ersten Aufruf einen definierten Wert besitzen — wie bei einem typischen Compiler.

In Wirklichkeit ist Comal keins von beiden — oder beides zur Hälfte, je nach Standpunkt. Das Handbuch spricht von einem »Three Pass Interpreter«, was möglicherweise der Wahrheit am nächsten kommt. Jedenfalls arbeitet Comal tatsächlich in drei Phasen. Die erste Phase kennen Sie bereits, wenn Sie schon Programmierversuche in Comal hinter sich haben. Es ist der Syntax-Check, der unmittelbar nach Eingabe einer Zeile ausgeführt wird. In dieser Phase wird die Programmzeile — ähnlich wie bei Basic — in eine kompakte Form umgewandelt, indem die Schlüsselwörter in Ein-Byte-Abkürzungen, sogenannte Token, umgewandelt werden.

Der Syntax-Check funktioniert im Prinzip recht einfach. Bei jeder eingegebenen Zeile wird zunächst überprüft, ob die Zeilennummer im erlaubten Bereich von 1 bis 9999 liegt. Anschließend wird getestet, ob eine Kommentarzeile vorliegt. Ein Kommentar wird in Comal allerdings nicht mit »REM« eingeleitet, sondern mit zwei Schrägstrichen. Jede Zeile, die mit »//« beginnt, wird daher nicht weiter beachtet. Alle anderen Zeilen durchlaufen jedoch die Routine »Text in Token wandeln«. Dabei wird ganz einfach überprüft, ob das erste Zeichen in der so codierten Zeile ein Token ist. Ein Token erkennt das Comal-System daran, daß in dem betreffenden Byte Bit 7 gesetzt ist; das funktioniert also völlig analog zu Basic.

Ist das erste Zeichen einer Zeile also weder das Kommentarsymbol noch ein Token, dann wird ein Syntax-Fehler gemeldet. Des weiteren wird einfach die Anzahl der öffnenden und schließenden Klammern einer Zeile gezählt. Ergibt sich eine Ungleichheit, dann resultiert das in einer Fehlermeldung. Eine kontextabhängige Syntaxprüfung findet jedoch nicht statt, das heißt, daß die Eingabezeile in dieser ersten Phase ohne Bezug zum Rest des Programms überprüft wird.

Sie können das Prinzip leicht selbst testen, indem Sie beispielsweise die folgende Zeile eintippen:
100 print sqr(2)

Wie zu erwarten, wird die Zeile widerspruchslos angenommen. Ändern Sie die Zeile jetzt doch einmal in
100 print xyz(2)

So funktioniert der Syntax-Check

Auch diese Zeile wird widerspruchslos angenommen, da sie mit einem zulässigen Schlüsselwort (print) beginnt und die gleiche Anzahl öffnende wie schließende Klammern enthält. Warum erfolgt hier keine Fehlermeldung ? Es gibt doch gar keine Funktion »xyz(2)«. Wirklich nicht? Was wäre, wenn Sie vor dem Eintippen der Zeile kein »NEW« gegeben hätten, um ein eventuell vorher vorhandenes Programm zu löschen? Woher wollten Sie dann mit Bestimmtheit sagen können, daß es eine Funktion xyz nicht gibt? Außerdem handelt es sich möglicherweise gar nicht um eine Funktion, sondern um ein eindimensionales Feld. Sie merken schon, solange Sie nur diese eine Zeile kennen und vom eventuell vorhandenen übrigen Programm keine Ahnung haben, sind Sie in der gleichen Situation wie der Comal-Interpreter. Der behandelt nämlich bei einer Eingabe auch nur diese eine Zeile und beachtet den Rest des Programms nicht weiter. Daher gibt er in so einem Fall auch lieber keine Fehlermeldung aus, denn möglicherweise wurde vorher im Programm eine Funktion xyz definiert oder ein Feld xyz dimensioniert. Die eingegebene Zeile ist also in Wahrheit tatsächlich syntaktisch korrekt, denn auf eine Print-Anweisung kann ein beliebiger Ausdruck folgen, es muß nicht unbedingt eine der Standardfunktionen oder eine Konstante sein.

Die Überprüfung, ob alle Programmzeilen zusammen auch tatsächlich ein vernünftiges Programm bilden, erfolgt in einer zweiten Pha-

Comal

se. Diese Phase wird mit dem Befehl »RUN« gestartet. Im Gegensatz zu Basic beginnt Comal nämlich nach »RUN« nicht unmittelbar mit dem Abarbeiten des Programms, sondern führt zunächst eine Überprüfung der Programmstruktur durch. Dabei wird zum Beispiel festgestellt, ob Schleifen richtig geschachtelt sind und beendet werden, ob jede aufgerufene Funktion und Prozedur auch tatsächlich definiert wurde und so fort. Bei der Gelegenheit werden gleich alle Sprungadressen berechnet und in eine parallel zum Programmtext angelegte Tabelle eingetragen.

Der Begriff »Sprungadresse« ist hier im weiteren Sinn zu verstehen, denn obwohl Comal über ein GO-TO-Statement verfügt, sollte dieses im Sinne einer übersichtlichen Programmstruktur nur in Ausnahmefällen verwendet werden. Mit Sprungadressen sind hier also auch »interne« Sprünge gemeint, zum Beispiel auf den Anfang einer Prozedur, wenn diese aufgerufen wird. Damit braucht der Comal-Interpreter beim späteren eigentlichen Programmlauf beispielsweise nicht mehr den gesamten Programmtext nach der Definition einer Prozedur oder Funktion zu durchsuchen, sondern findet die entsprechende Adresse viel schneller durch »Nachschlagen« in einer Tabelle.

Fehlerhafte Programme werden gar nicht ausgeführt

Dieser Vorgang der Ersetzung von symbolischen Adressen (Prozedur-, Funktions-, Variablen- und Labelnamen) durch die tatsächlichen Adressen dieser Objekte im Speicher ist eine der wesentlichen Aufgaben eines Compilers. Insofern hat Comal also tatsächlich Compiler-Eigenschaften. Allerdings werden zum Beispiel arithmetische Ausdrücke nicht in eine andere, maschinennahe Form übersetzt, wie das bei einem »richtigen« Compiler der Fall wäre. Auch findet keinerlei Übersetzung in Maschinensprache statt, so daß es tatsächlich falsch wäre, von einem Compiler zu sprechen.

Nachdem alle Überprüfungen und Übersetzungen abgeschlossen sind, beginnt schließlich die dritte und letzte Phase der Interpretation, nämlich der eigentliche Programmlauf. Diese Phase wird automatisch eingeleitet, wenn in Phase zwei kein Fehler aufgetreten ist. Für den Benutzer sind die Phasen zwei und drei daher in der Regel nicht zu unterscheiden. Daß es sie aber wirklich gibt, kann man mit einem kleinen Testprogramm sehr einfach feststellen:

```
10 print "hier ist Zeile 10"
20 //
30 // es folgt ein Fehler
40 // in der Programmstruktur
50 //
60 endif
```

In Zeile 60 steht das Schlüsselwort für das Ende eines If-Blocks, aber nirgends vorher taucht ein »if ... then« auf.

Ein reiner Interpreter — wie Basic — würde nach »RUN« die Meldung »hier ist Zeile 10« auf den Bildschirm schreiben und erst anschließend feststellen, daß hier ein endif ohne if vorliegt. Lassen Sie aber dieses kleine Programm einmal in Comal laufen und sehen Sie selbst, was passiert. Nach »RUN« wird die Print-Anweisung in Zeile 10 nicht ausgeführt, sondern erst einmal Phase zwei gestartet und das Programm überprüft. Dabei wird der Fehler gefunden, und es erscheint eine entsprechende Meldung. Phase drei, nämlich die eigentliche Programmausführung, wird wegen dieses Fehlers gar nicht erst erreicht.

Phase drei wird überhaupt nur gestartet, wenn ein Hauptprogramm vorhanden ist. Besteht der gesamte Programmtext nur aus Funktions- oder Prozedurdefinitionen, dann wird nach »RUN« ebenfalls nichts ausgeführt. Allerdings können nun — und das ist ganz wesentlich — alle im Programm definierten Funktionen und Prozeduren im Direktmodus aufgerufen werden.

Das ist eine Eigenschaft von Comal, die für den Benutzer von großer Wichtigkeit ist. Denn damit kann, wie sonst in dieser Form nur bei Forth, Logo oder Lisp möglich, der Sprachumfang fast beliebig erwei-

Comal-Befehl	Bedeutung
//	Kommentarzeile
AUTO	automatische Zeilennumerierung
AUTO 100,5	numeriert ab 100 in Fünferschritten
BASIC	Rückkehr ins Basic
CAT	Directory listen, ohne Programmzerstörung
CHAIN "name"	Laden und Starten von Programmen
DEL	Zeilen löschen
DEL 10-30	löscht Zeilen 10 bis 30
DELETE "name"	löscht ein File auf der Disk
EDIT	listet Programm ohne Zeileneinrückung
EDIT 10-30	editieren der Zeilen 10 bis 30
ENTER "name"	sequentielles Programmfile laden
LIST	listet Programm strukturiert
LIST 10-30	listet Zeilen 10 bis 30
LIST "name"	speichert Programm als sequentielles File
LOAD "name"	lädt Programm von Diskette
NEW	löscht Arbeitsspeicher und Variable
PASS "string"	sendet einen Kommandostring an die Floppy
RENUM	Programm in Zehnerschritten neu numerieren
RENUM 100,5	numeriert ab Zeile 100 in Fünferschritten
RUN	startet Programmlauf
SAVE "name"	Programm abspeichern
SELECT	Ausgabegerät wählen
SELECT "LP:"	Ausgabegerät Drucker
SELECT "DS:"	Ausgabegerät Bildschirm
STATUS$	enthält Status des Disk-Kanals
STATUS	Abkürzung für PRINT STATUS$

Tabelle 1. Wichtige Comal-Kommandos und Editierbefehle

tert werden. Benötigen Sie zum Beispiel eine Funktion, ähnlich wie PEEK, die aber nicht nur ein Byte, sondern ein 16-Bit-Wort aus dem Speicher liest? Kein Problem. Geben Sie im Direktmodus NEW ein und danach die folgenden Programmzeilen (Sie können sich das Tippen der Zeilennummern sparen, wenn Sie zuvor den Befehl »AUTO« eingeben):

```
10 func deek(x)
20 wert:=peek(x)+256*peek(x+1)
30 return(wert)
40 endfunc deek
```

Bei diesem Vierzeiler handelt es sich um die Definition einer Funktion »DEEK« mit einem Parameter. In Zeile 20 werden die nötigen Berechnungen ausgeführt. Das Schlüsselwort »RETURN« in Zeile 30 hat eine andere Bedeutung als in Basic. Es besagt, daß die Funktion als Ergebnis des Funktionsaufrufes den Wert der Variablen »WERT« zurückliefern soll. Mit »ENDFUNC DEEK« schließlich wird dem Comal-Interpreter das Ende der Funktionsdefinition angezeigt. Nach »RUN« erfolgt sofort die Meldung »END AT 0040«, und der Comal-Sprachschatz ist um die Funktion »DEEK« erweitert. »PRINT DEEK(209)« zeigt beispielsweise die Speicheradresse des Beginns der aktuellen Bildschirmzeile an.

Doch damit zunächst einmal genug über die Arbeitsweise des Comal-Interpreters. Wenden wir uns nun einigen wichtigen Kommandos zu, die das Arbeiten mit Diskette, Kassette und Drucker ermöglichen.

Wie kommt das Programm auf die Diskette?

Nehmen wir einmal an, wir hätten gerade unser erstes kleines Testprogramm in Comal geschrieben. Natürlich wollen wir unser Erstlingswerk gerne der Nachwelt erhalten. Doch wie bekommen wir das Programm auf die Diskette (oder auch auf die Kassette)? Etwa mit SAVE, wie in Basic? Ja, genauso.

Wie bereits zu Anfang erwähnt, ist Comal stark an Basic angelehnt (siehe Tabelle 1). Die Befehle »LOAD« und »SAVE« dienen zum Laden und Speichern von Programmen. »VERIFY« steht leider in der Comal Version 0.14 noch nicht zur Verfügung. Dafür gibt es aber zusätzlich den »CHAIN«-Befehl, der ein Programm von Diskette lädt und automatisch startet. Im Unterschied zu Basic ist Comal allerdings diskettenorientiert, das heißt Sie können sich das lästige »,8« sparen. Soll allerdings statt auf Diskette auf ein Kassettenlaufwerk zugegriffen werden, so muß die Sekundäradresse 1 angegeben werden.

Zur einfacheren Bedienung der Floppy ist in Comal der Befehl »PASS« vorgesehen, der einen Kommandostring an die Floppy sendet. Soll zum Beispiel eine Diskette neu initialisiert werden, dann braucht nicht erst umständlich der Kommandokanal geöffnet werden, sondern es reicht der Befehl »PASS "I" «. Die Systemvariable »STATUS$« enthält immer den Fehlerstatus der Floppy, und zwar im Klartext. Wem das Eintippen von »PRINT STATUS$« noch zu mühselig ist, der kann auch nur »STATUS« eintippen. Comal versteht dann schon, was gemeint ist.

Erheblich komfortabler als in Basic ist auch das Laden des Directory. Einfach »CAT« (für catalog) eingeben, und das Inhaltsverzeichnis der Diskette wird angezeigt, und zwar ohne das Programm zu zerstören.

Neben »LOAD« und »SAVE« gibt es noch zwei weitere Lade- und Speicherbefehle. »LIST "name" « legt ein Comal-Programm als sequentielles File auf Diskette ab. Mit »ENTER "name" « wird ein solches sequentielles File wieder als Comal-Programm eingelesen.

Was ist der Unterschied zwischen diesen beiden Formen des Abspeicherns? Die Antwort darauf mag der eine oder andere schon von Basic her kennen. Der Comal-Befehl »LIST "name" « hat die gleiche Wirkung wie die Basic-Befehlsfolge »OPEN 1,8,1, "name,s,w" « : CMD 1 : LIST : CLOSE 1«. Anschließend existiert in beiden Fällen auf der Diskette ein sequentielles File, das das entsprechende Basic- oder Comal-Programm als reine ASCII-Zeichenfolge, also ohne Token, enthält.

Das kann durchaus nützlich sein, weil man ein solcherart gespeichertes Programm sehr leicht und ohne Kenntnis der speziellen Befehlstoken weiter bearbeiten kann. Auf der Comal-Diskette befindet sich zum Beispiel ein Demo-Programm »FORMATTER.COM«, mit dessen Hilfe sich solche sequentiellen Files formatiert auf dem Drucker listen lassen, wobei alle wichtigen Parameter eingestellt werden können.

Zum Laden mit »ENTER« gibt es kein einfaches Basic-Äquivalent, sondern dort muß man schon ein kleines Programm schreiben. In Comal dient der »ENTER«-Befehl dazu, ein sequentielles File zu lesen und dabei wieder in »normalen« Comal-Text mit Token-Codierung umzuwandeln.

```
0010 // Turtle-Grafik Demo (ev140984)
0020 //
0030 //
0040 // Quadratseite zeichnen
0050 //
0060 proc seite(laenge)
0070  forward laenge
0080  left 90
0090 endproc seite
0100 //
0110 // Quadrat zeichnen
0120 //
0130 proc quadrat(laenge)
0140  for i:=1 to 4 do
0150   seite(laenge)
0160  endfor i
0170 endproc quadrat
0180 //
0190 // === Hauptprogramm ===
0200 //
0210 setgraphic 0 // Hires ein
0220 clear  // Hires loeschen
0230 for laenge:=5 to 50 step 5 do
0240  turtlesize 5
0250  quadrat(laenge)
0260 endfor laenge
0270 end
```

Eine kleine Demonstration der Turtle-Grafik

Natürlich wäre es einigermaßen lästig, wenn man Programmlistings nur auf solchen verschlungenen Pfaden auf dem Drucker ausgeben könnte. Aber Comal wäre nicht Comal, wenn nicht alles etwas einfacher ginge als in anderen Sprachen. Da ein Umschalten zwischen Bildschirm- und Druckerausgabe oft gebraucht wird, gibt es in Comal einen speziellen Befehl dafür: Mit »SELECT OUTPUT "gerät" « kann das gewünschte Ausgabegerät gewählt werden. Die Angabe des Schlüsselwortes »OUTPUT« ist dabei nicht unbedingt notwendig. Für »gerät« gibt es zwei Möglichkeiten: »LP:« (Line Printer) für Druckerausgabe, »DS:« (Data Screen) für Ausgabe auf den Bildschirm.

Um also ein Listing auf den Drucker auszugeben, gibt man nacheinander die beiden folgenden Befehle ein:

```
SELECT "LP:"
LIST
```

Nach dem LISTen wird übrigens automatisch wieder auf Bildschirmausgabe umgeschaltet.

Damit sind alle wichtigen Befehle behandelt worden, um Comal-Programme vernünftig editieren zu können. In der nächsten Folge werden wir uns (endlich) dem eigentlichen Programmieren in Comal zuwenden. (ev)