Wegweiser zum Z-System

von Jörg Linder

Ein Red Nil Design *Dokument*

Einleitung

Angeregt durch meine Artikelreihe in den KC-News sind inwischen einige Mitglieder des KC-Clubs neugierig geworden und haben sich das Z-System (in Form von NZ-COM) zugelegt. Einer der größten Vorteile des Z-Systems – die Flexibilität – kehrt sich für den Neueinsteiger leicht ins Gegenteil um. Zu verwirrend ist die Vielzahl und die Funktion der Systemmodule. In meinen Artikeln habe ich zwar versucht, ein wenig Licht ins Dunkel zu bringen, aber letztendlich muß sich jeder selbst „durchkämpfen".

Um die Anfangsschwierigkeiten etwas zu verringern, soll innerhalb dieser Projektgruppe Hilfestellung gegeben werden. Die Gruppe ist jedoch nicht nur für Anfänger gedacht, sondern auch für Fortgeschrittene, die das Z-System und dessen Fähigkeiten voll ausschöpfen wollen. Das Z-System ist derart flexibel und leistungsfähig, daß selbst langjährige Benutzer neue Anwendungsgebiete entdecken (können).

Ohne Environment läuft nichts!

Sicherlich könnte man viele Elemente des Z-Systems als „Grundbaustein“ bezeichnen, doch nichts ist so wichtig wie der Environment Descriptor. Auf die in diesem Datenbereich gespeicherten Informationen greifen zahlreiche Routinen und Systembestandteile zu. Daher soll er näher betrachtet werden.

Zunächst einmal bedeutet Environment Descriptor nichts anderes als „Beschreibung der Umgebung“. Hier sind also Informationen über die Systemumgebung in einer genau definierten Reihenfolge gespeichert. Der Environment Descriptor ist exakt 128 Bytes (= 1 Record) groß und beinhaltet neben den Angaben zu den Systemelementen (Adresse und Größe von RCP, FCP, IOP usw.) z. B. auch die Taktfrequenz und die Anzahl der Bildschirmzeilen bzw. -spalten.

Weil die Angaben von NZ-COM zum Aufbau eines Systems benötigt werden, muß zuvor ein gültiger Descriptor erstellt werden. Dies erledigt man mittels MKZCM. Für den Einsteiger bedeutet es jedoch, daß die Eigenschaften des Systems bereits festgelegt werden müssen, wenn keine oder nur geringe Vorkenntnisse über die Funktionsweise der Systemelemente vorhanden sind. Glücklicherweise kann aber jederzeit ein neuer Descriptor (für ein System mit anderen Elementen) erstellt werden, auf dessen Grundlage NZ-COM ein neues/anderes System lädt.

Der Environment Descriptor wird aber nicht nur während des Ladevorganges benötigt, sondern auch für die Z-System-Programme und die Systemelemente. Deshalb ist er selbst Bestandteil der Umgebung und steht permanent zur Verfügung. Programme können unter Zuhilfenahme von Routinen aus der Z3LIB den Descriptor ausfindig machen und anschließend auf die enthaltenen Informationen zugreifen.

Eine Z-System-Umgebung ist zwar kompatibel zum originalen CP/M, jedoch sollten Programme im Bedarfsfall die Adressen von BIOS, BDOS und CCP dem Environment Descriptor entnehmen und nicht auf berechnete Werte zurückgreifen.

Von MKZCM werden zwei Descriptor-Dateien erstellt, die den gleichen Inhalt jedoch in unterschiedlichen Formaten haben. Die ENV-Datei entspricht exakt dem Descriptor, wie er später resident im Speicher steht. Leichter editierbar ist die ZCM-Datei. Sie enthält die Marken mit den dazugehörigen Werten im Klartext. Mit Hilfe einer Textverarbeitung oder eines einfachen Editors können Änderungen schnell vorgenommen werden.

So empfiehlt es sich z. B. den Laufwerksvektor nachträglich anzupassen. Grundsätzlich wird nämlich von einem voll ausgebauten CP/M System mit 16 Laufwerken (A bis P) ausgegangen. Der Laufwerksvektor ist ein 16-Bit-Wort, in dem jedes Bit stellvertretend für ein Laufwerk steht (siehe Handbuch). Es ist also auch möglich, „Lücken zwischen den Laufwerken“ zu berücksichtigen. Gute Z-System-Programme verwenden den Laufwerksvektor bei Diskettenzugriffen. Dadurch kann so manches Problem beim versuchten Zugriff auf ein nicht vorhandenes Laufwerk erspart bleiben.

Selbstverständlich können auch die anderen Werte in der ZCM-Datei den eigenen Wünschen angepaßt werden (z. B. Anzahl der Bildschirmzeilen). Grundsätzlich lädt NZ-COM (entsprechend der im Handbuch beschriebenen Hierarchie) die ZCM-Datei zuerst, doch um Verwirrung zu vermeiden, sollte eine gleichnamige ENV-Datei gelöscht werden.

Auf verschlungenen Wegen – Suchpfade

Erst mit dem Z-System habe ich unter CP/M den Komfort eines Suchpfades kennengelernt, den ich jetzt nicht mehr missen möchte. Durch die revolutionären Eigenschaften von ZSDOS/ZDDOS tut sich nun eine ganze Welt von Möglichkeiten auf, die bei sinnvoller Anwendung die Benutzung von Programmen aus nahezu jedem Verzeichnis ermöglicht.

Doch erst einmal zum Kommandosuchpfad von NZ-COM. Wie dem Namen schon andeutungsweise zu entnehmen ist, werden entlang dieses Pfades Kommandodateien (Typ COM) gesucht. Mit anderen Worten: Alle Programme, die sich in Verzeichnissen entlang des Pfades befinden werden vom Kommandoprozessor unabhängig vom aktuellen Verzeichnis gefunden und können somit ohne Wechsel in das jeweilige Verzeichnis ausgeführt werden.

Der Kommandosuchpfad kann aus maximal 5 Elementen bestehen, wobei dem letzten eine besondere Bedeutung zukommt. Mit dem letzten Pfadelement wird das sogenannte Root-Verzeichnis (Root = Wurzel) bestimmt. Spezielle Programme und der CPR selbst erwarten in diesem Verzeichnis ihre Dateien. So wird der erweiterte Kommandoprozessor nur gefunden, wenn er sich als Datei namens CMDRUN.COM in diesem Verzeichnis befindet. Ebenso suchen ZFILER, ARUNZ und andere Tools ihre Dateien dort.

Auch NZ-COM benutzt das Root-Verzeichnis. Während des Startvorgangs werden dort zwei Dateien angelegt. Zum einen handelt es sich um NZCOM.CCP, in dem sich der jeweils aktuelle Kommandoprozessor befindet (also nie löschen!!!), und zum anderen um NZCPM.COM. Letztgenanntes Programm beendet das Z-System und ermöglicht so die Rückkehr zum ursprünglichen System. Die letztgenannte Datei wird jedoch nur angelegt, wenn sie beim Start noch nicht vorhanden war. Hat sich das Ausgangssystem geändert, so muß diese Datei gelöscht werden, um eine aktuelle Version beim nächsten Start des Z-Systems erstellen zu lassen.

Eine Ebene tiefer arbeitet der DOS-Suchpfad von ZSDOS. Dieser kann maximal 3 Elemente enthalten. Im Gegensatz zum Kommandosuchpfad des CPR werden hier nicht nur Programme (Typ COM) gefunden, sondern alle Dateien. Es können also auch Daten- oder Overlaydateien geladen werden, die sich in Verzeichnissen entlang des Pfades befinden, ohne in das jeweilige Verzeichnis wechseln zu müssen. Dies ist insbesondere für Programme wie WordStar oder dBase interessant, da sie über den DOS-Pfad auf ihre Overlays zugreifen können. Erst durch diese Eigenschaft wird die Benutzung dieser Programme unabhängig vom aktuellen Verzeichnis ermöglicht.

Wegen der besonderen Bedeutung sollten die Verzeichnisse des DOS-Pfades mit Bedacht gewählt werden. Insbesondere die Beschränkung auf lediglich 3 Verzeichnisse macht die Wahl besonders schwer. ZSDOS bzw. ZDDOS bieten jedoch eine weitere Zugriffsart, die in Kombination mit dem DOS-Pfad nahezu unbegrenzte Möglichkeiten bietet – öffentliche Dateien.

Während die Pfadzugriffe an Kombinationen aus Laufwerk und Nutzerbereich gebunden sind, wird der öffentliche Zugriff über ein Dateiattribut gesteuert und ist somit direkt mit dem Datenträger (Medium) bzw. der Datei verbunden. Öffentlich gemachte Dateien werden beim Zugriff auf das Laufwerk unabhänging von ihrem Nutzerbereich gefunden. Zur Vermeidung von Konflikten muß der Name einer öffentlichen Datei auf dem jeweiligen Datenträger einzigartig sein. Durch eine gut gewählte Anordnung der Elemente

des Kommandosuchpfades und des DOS-Pfades sowie der öffentlichen Dateien lassen sich die Möglichkeiten des erweiterten Zugriffes optimal ausnutzen.

Bei der Arbeit mit dem Z-System hat sich folgende Verfahrensweise bewährt: Als Root-Verzeichnis wird ein hoher Nutzerbereich der RAM-Floppy (A15:) benutzt. Dadurch wird gewährleistet, daß die von NZ-COM angelegten Dateien stets aktuell sind. Häufig benötigte Tools und der erweiterte Kommandoprozessor (im Normalfall ARUNZ) werden mittels Startscript in das Root-Verzeichnis kopiert. Alle anderen Z-System-Programme befinden sich in einem hohen Nutzerbereich der Festplatte (C15:). Für Help-Dateien sollte ein eigener Nutzerbereich auf der Festplatte reserviert werden (C14:).

Elementare CP/M-Programme werden in C0: gespeichert; größere CP/M-Programme wie WordStar oder dBase erhalten einen eigenen Nutzerbereich auf der Festplatte (C1:, C2: usw.). Außerdem erhalten diese Programme ebenso wie ihre Overlay-Dateien den Status von öffentlichen Dateien. Somit können sie von jedem anderen Verzeichnis aus benutzt werden. Wer auf Nummer sicher gehen will, kann den Dateien zusätzlich das Attribut R/O (Schreibschutz) geben.

Die anderen Partitionen der Festplatte werden entweder als Datenspeicher (Laufwerk D:) oder vorrangig für CAOS (Laufwerk E:) genutzt. Selbstverständlich können auch Verzeichnisse dieser Laufwerke im Pfad angegeben werden. Auf keinen Fall sollte ein Verzeichnis eines Diskettenlaufwerkes in einen der Pfade aufgenommen werden, da es wegen der großen Zugriffszeit die Arbeit bremst.

Nach dem Start wird als Arbeitsverzeichnis A0: benutzt. Dies hat den Vorteil, daß Programme benutzt und ausprobiert werden können, ohne Datenmüll auf der Festplatte zu hinterlassen. Wichtige Daten müssen aber vor dem Ausschalten gesichert werden!

Hier nochmal eine kurze Zusammenfassung:

Kommandosuchpfad:	A15 C15 A0 $$ A15
DOS-Suchpfad:	C0 D0 E0, Pfad-Verzeichniszugriff
öffentliche Dateien:	aktiv

Dazu passend die Namen der Verzeichnisse (NDR):

A0:WORK	A15:ROOT			
C0:CPM	C1:WORDSTAR	C2:DBASE	C14:HELP	C15:COMMANDS
E0:CAOS				

IF, ELSE, ENDIF oder Was wäre wenn..?

So flexibel und leistungsfähig wie das Z-System und seine Programme auch sein mögen, es gibt mindestens einen Anwendungsfall, den das Lieblingstool nicht berücksichtigt. Zum Beispiel soll bei Vorhandensein einer bestimmten Datei ein spezielles Programm ausgeführt werden, ansonsten soll eine Meldung auf dem Bildschirm ausgegeben werden.

Genau diesen Bereich deckt ein unscheinbares Element des Z-Systems ab – der FCP. Laut Handbuch heißt FCP „Flow Command Package", zu deutsch Fluß Steuerung. Dies bringt uns erstmal auch nicht viel weiter. Die Anmerkung im Handbuch, daß der Umgang mit der Fluß Steuerung gewöhnungsbedürftig und mit einem gewissen Lernaufwand verbunden ist, kann ich nur bestätigen. Viel zu spät entdeckt man den Nutzen von IF, ELSE & Co.

Hat man die ersten „Gehversuche" mit den FCP-Befehlen gemacht, fehlt wahrscheinlich immer noch das Verständnis. Daß sämtliche Befehle ignoriert werden, solange der IF-Status „falsch" ist, bekommt man recht schnell heraus. Auch der Statuswechsel mittels ELSE oder das Rücksetzen durch ZIF funktioniert reibungslos. Doch wie sollen diese Dinge bei der täglichen Arbeit mit dem Z-System behilflich sein?

Am besten lernt man den Umgang mit der Fluß Steuerung, wenn man sich Beispielscripts und -makros ansieht, die mit NZ-COM ausgeliefert werden. Denn erst in automatisierten Abläufen macht die Arbeit mit dem FCP Sinn. In einer mühsam eingetippten Kommandozeile ist der Aufwand normalerweise zwecklos.

Da ich auf Scripts erst später eingehen werde, möchte ich an dieser Stelle nur die FCP-Befehle behandeln. Standardmäßig wird ein FCP mit einer ganzen Reihe von Befehlen installiert. Eine Bedingung wird mit dem Befehl IF (deutsch: Wenn) eingeleitet. Nur wenn die Bedingung erfüllt (also „wahr") ist, werden die nachfolgenden Befehle ausgeführt. Anderenfalls werden sie bis zum Eintreten des Status „wahr" ignoriert. Es können auch mehrere IF-Bedingungen in bis zu acht Ebenen verschachtelt werden. Es gilt dann der Status der jeweils letzten Ebene.

Die Bedeutung der logischen Operatoren AND und OR zur Verknüpfung von Bedingungsparametern dürfte klar sein. Der Befehl ELSE (deutsch: Sonst) kehrt den aktuellen Status um. Somit kann auch auf eine nicht erfüllte Bedingung reagiert werden. Um eine Bedingung(sebene) abzuschließen, benutzt man den Befehl FI („IF" rückwärts). Anschließend wird die vorhergehende Ebene aktiv bzw. nach der letzten IF-Ebene die Fluß Steuerung beendet. Hat man sich zu sehr in den Ebenen „verstrickt" oder sollen alle Ebenen zugleich beendet werden, verwendet man am besten den Befehl ZIF (Zero IF). Damit wird die Fluß Steuerung augenblicklich beendet (Null-Status).

Ebenso wie bei fast allen Elementen des Z-Systems hat man auch beim FCP die Möglichkeit, zwischen unterschiedlichen Varianten zu wählen. Analog zur Leistungsfähigkeit verhält sich dann auch die Größe des FCP – bei minimalen 4 Records beginnend gibt es nach oben kaum Beschränkungen. Doch selbst die umfangreichsten FCPs können nicht die Leistungsfähigkeit von IF.COM bieten. Es ist sozusagen die Programmvariante des Systemelementes und sollte in keinem Root-Verzeichnis fehlen.

IF.COM wird aufgerufen, sobald ein Kommando zur Fluß Steurung intern nicht gefunden werden kann. Dies ist normalerweise bei Befehlen der Fall, die die Existenz einer Datei oder den Inhalt der User-Register testen. Zwar dauert der Zugriff auf IF.COM länger als auf die internen Befehle, aber es kann sich trotzdem

lohnen, einen Großteil der Fluß Steuerung aus dem FCP (und damit aus dem Arbeitsspeicher) zu verbannen, da die meiste Arbeit mit nur 4 Befehlen (IF, ELSE, FI, ZIF) erledigt werden kann.

Hierzu ein Beispiel: Das Programm VIEW ermöglicht die Betrachtung von PIP/PIF- und HIP/HIF-Bildern auf dem KC unter CP/M bzw. MicroDOS. Standardmäßig wird dabei von der Standardauflösung – also von PIP/PIF – ausgegangen. Soll ein HiRes-Bild angezeigt werden, so muß nach dem Dateinamen zusätzlich ein „H“ angegeben werden. Die korrekte Syntax lautet:

```
VIEW [dr:]Dateiname [H]
```

Um Bilddateien aus ZFILER heraus betrachten zu können, muß zuvor der Dateityp überprüft und eine entsprechende Kommandozeile generiert werden. Das ZFILER-Makro könnte wie folgt aussehen:

```
IF $ft=PIP;VIEW $fn;ELSE;IF $ft=HIP;VIEW $fn H;ELSE;ECHO Keine Bilddatei!^G;ZIF
```

In einer etwas strukturierteren Form würde sich die Kommandozeile so lesen:

```
Wenn Dateityp = PIP
    dann lautet die Kommandozeile: VIEW Dateiname
Sonst
    Wenn Dateityp = HIP
        dann lautet die Kommandozeile: VIEW Dateiname H
    Sonst
        Ausgabe der Meldung: Keine Bilddatei!
        (Control-G wird in akustisches Signal umgewandelt)
Wenn-Status auf Null
```

Wie bereits erwähnt, macht die Fluß Steuerung auf Kommandozeilenebene wenig Sinn. Zum einen kann man nach einem kurzen Blick aufs Directory selbst entscheiden, um welchen Dateityp es sich handelt und zum anderen stehen spezielle Platzhalter (z. B. für Komponenten des Dateinamens) nur unter Umgebungen wie ZFILER oder ARUNZ zur Verfügung.

Alias, Makro, Script und andere Raffinessen

Bereits im Zusammenhang mit der Fluß Steuerung kamen andeutungsweise die Vorzüge von Makros zur Sprache. Vornehmlich wiederkehrende Arbeitsschritte lassen sich unter dem Z-System automatisieren. Die vom Benutzer einzugebenden Befehle oder Aktionen lassen sich auf ein Minimum reduzieren. Anfangs tut man sich ein wenig schwer, ein Makro für ZFILER oder ein ARUNZ-Script zu „programmieren", doch macht sich die Mühe schnell bezahlt, wenn stets dieselben Befehle einzugeben sind.

So unterschiedlich die zur Verfügung stehenden Möglichkeiten für automatisierte Arbeitsabläufe sind, so verschieden ist auch der Komfort der jeweiligen Programme. Am besten fangen wir mit der einfachsten Software an (SALIAS) und arbeiten uns dann zur komplexen vor (ARUNZ).

SALIAS ist ein komfortables Programm zur bildschirmorientierten Erstellung eines Alias. Laut Handbuch ist das ein einzelnes Wort bzw. ein Befehl, der stellvertretend für einen längeren oder zusammengesetzten Befehl steht. Konkret heißt das also, daß unter einem Kommando eine Reihe von Befehlen zusammengefaßt werden. Diese werden nacheinander abgearbeitet, wenn der Aliasname (das Kommando) aufgerufen wird. Dadurch kann man sich also eine ganze Menge Tipperei ersparen!

Nach dem Aufruf von SALIAS erscheint die einer Textverarbeitung ähnliche Oberfläche. Die oberste Bildschirmzeile fungiert als Statuszeile, in der vor allem die Angabe von „Free" sehr interessant ist, denn dort werden die noch zur Verfügung stehenden Zeichen angezeigt. Nach dem Eingabebereich folgt in der untersten Bildschirmzeile die Befehlszeile. Drückt man ESC, dann werden dort die möglichen Aktionen angezeigt, die über den jeweiligen Großbuchstaben ausgeführt werden.

Während des Editiermodus können nahezu alle von WordStar (bzw. TPKC) bekannten Control-Codes benutzt werden. Im Zweifelsfall hilft die Eingabe von Control-J weiter, die eine kurze Auflistung aller Codes zeigt. Die von SALIAS erzeugten Aliasdateien sind ausführbare Programme und erhalten den Dateityp COM und sind somit von „echten" Programmen nicht zu unterscheiden. Wird ein derartiger Alias aufgerufen, so geschieht folgendes: Der im Alias enthaltene Programmcode lädt die eingegebenen Befehle (durch Semikolon getrennt) in den Mehrfach-Kommandozeilenpuffer. Anschließend wird die Kontrolle wieder an den Kommandoprozessor übergeben, der dann diese (Mehrfach-)Kommandozeile ausführt.

Davon bekommt man aber als Anwender nichts mit. Man kann sich beruhigt zurücklehnen und dem Rechner bei der Abarbeitung der Befehle zusehen. Leider hat diese sehr einfache und bequeme Variante ein paar Nachteile. Zum einen wird für jeden Alias Disketten- und Directoryplatz beansprucht. Auf einer Fesplatte mit 4k-Blöcken werden für die kleinen Dateien jeweils 4 kB belegt (ganz zu schweigen von den sowieso knappen Directory-Einträgen). Außerdem hat man mit SALIAS keinen Einfluß auf bestimmte Parameter wie z. B. Dateinamen/-typ oder Verzeichnis. Der Alias wird so abgearbeitet wie eingegeben; es kann keine Rücksicht auf Spezialfälle genommen werden.

Ganz anders sieht es da mit den ZFILER-Makros aus. Durch die Verbindung mit der zweifelsohne komfortablen Shell „ZFILER" bieten sich ganz besondere Vorzüge. So ist es möglich, die zu be-/verarbeitenden Dateien nicht nur über eine Auswahl mit Hilfe der Jokerzeichen „?" und „*" auszuwählen, sondern individuell durch das Markieren in der angezeigten Dateiliste. Darüberhinaus bietet ZFILER bereits ein beachtliches Potential eigener Befehle für Dateioperationen (z. B. Kopieren, Löschen oder Betrachten). Die Makro-

datei muß sich im Root-Verzeichnis befinden, sonst kann sie ZFILER nicht finden und demzufolge auch nicht verwenden.

Im Gegensatz zu SALIAS werden hier nicht alle Makros in einzelnen Dateien gespeichert, sondern in der Datei ZFILER.CMD zusammengefaßt. Dies bietet neben der Tatsache, daß alle Makros gleich „auf einen Blick" verfügbar sind, auch die Vorteile der Einsparung von Disketten- und Directoryplatz. Da es sich bei ZFILER.CMD um eine normale Textdatei handelt, ist zur Bearbeitung ein Editor oder ein Textverarbeitungsprogramm notwendig (Nondocument-Modus).

Die Makrodatei hat einen einfachen Aufbau. Nach dem Anzeigebereich folgen die Makros, die später in den Kommandozeilenpuffer übertragen werden. Die Gestaltung des Anzeigebereiches steht dem Benutzer völlig frei. Diese Zeilen werden als eine Art Hilfeseite angezeigt, wenn man am ZFILER-Prompt ein „#" eingibt. Ein Makro wird über den zugeordneten Buchstaben aufgerufen. Dieser steht in der ersten Spalte einer Makrodefinition. Nach einem Leerzeichen folgt die auszuführende Kommandozeile. Mehrere Befehle werden entsprechend den ZCPR-Konventionen durch Semikolon voneinander getrennt. Gegenüber SALIAS bietet ZFILER den entscheidenden Vorteil von Platzhaltern. Spezielle Zeichenkombinationen werden als Symbol verwendet, das stellvertretend für Komponenten der jeweiligen Dateiangabe steht.

So wird z. B. $fn als Platzhalter für den Dateinamen der aktivierten Datei benutzt. Das ist bei einer einfachen Operation die Datei, auf die der Markierungspfeil zeigt. ZFILER-Makros können aber auch auf eine Dateigruppe angewendet werden. Bei einer Gruppenoperation wird der gewählte Befehl der Reihe nach für alle markierten Dateien ausgeführt, d. h. es wird nacheinander jeweils eine Datei aktiviert und der Befehl/Makro auf sie angewendet. Dadurch müssen die aufgerufenen Programme nicht unbedingt in der Lage sein, Dateinamen mit Jokerzeichen verarbeiten zu können.

Durch die verschiedenen Platzhalter kann man während der Makroprogrammierung Einfluß auf Spezialfälle nehmen und so bestimmte Fehler von vornherein ausschließen. Auch der Wechsel in ein bestimmtes Verzeichnis zur Ausführung des Makros und die anschließende Rückkehr zum Herkunftsverzeichnis läßt sich mittels Platzhalter bewerkstelligen. Wie man sieht, ist ein ZFILER-Makro um einiges mächtiger als ein normaler Alias, allerdings ist die Änderung der Datei ZFILER.CMD nicht so schnell erledigt wie die eines Alias. Erschwerend kommt hinzu, daß für den Einsteiger die kryptisch anmutenden Symbole der Platzhalter nicht gerade zum leichten Verständnis beitragen.

Die ZFILER-Makros haben aber neben den genannten Vorzügen auch einen entscheidenden Nachteil: Man muß ZFILER starten, um sie ausführen zu können. Das stellt im Normalfall kein Problem dar, ist aber mitunter hinderlich, wenn eine Aktion ausgeführt werden soll, die nicht dateibezogen ist oder Programme aufruft, die den Shell Stack löschen.

Mit den ARUNZ-Scripts erhält man die wohl komfortabelste und komplexeste Möglichkeit der Ablaufautomatisierung. Doch gerade die Komplexität „erschlägt" den Anwender und läßt ihn nur spärlich Fortschritte machen. Ist ein ARUNZ-Script jedoch einmal programmiert, gibt es nur selten Anlaß zu späteren Änderungen.

Im Normalfall wird ARUNZ als erweiterter Kommandoprozessor eingesetzt. Dazu muß sich das Programm unter dem Namen CMDRUN.COM im Root-Verzeichnis befinden; ebenso die Datei ALIAS.CMD mit den Scripts. Alle Befehle, die weder in den internen Systempaketen noch als COM-Datei entlang des Komman-

dosuchpfades gefunden werden können, werden dem erweiterten Kommandoprozessor übergeben. Durch die Angabe von Jokerzeichen in Dateibezeichnungen können Scripts auf mehrere Dateien angewendet werden (sofern die aufgerufenen Programme dies unterstützen). Eine derart komfortable Möglichkeit, Dateien wie in ZFILER als Gruppen zu markieren, besteht leider nicht.

Doch dafür bietet ARUNZ eine fast unüberschaubare Vielfalt an Platzhaltern für (fast) alle Anwendugnsbereiche. Die Platzhalter bestehen hier nicht nur aus einigen festgelegten Symbolen, sondern sie werden aus verschiedenen Parametern kombiniert. Daraus resultiert ein immenser Funktionsumfang. Darüberhinaus bietet ARUNZ sogar den Zugriff auf Bestandteile des Z-Systems (z. B. User-Register, Adressen der Systemelemente) und auf den Namen des Scripts.

Eine wichtige Eigenschaft soll hier nicht unerwähnt bleiben, die sowohl ein von SALIAS erzeugter Alias als auch ein ARUNZ-Script hat. Beide Arten können rekursiv gestaltet werden, d. h. sie können sich selbst aufrufen. Allerdings muß dabei beachtet werden, daß der Name des Alias oder des Scripts (also der eigene Wiederaufruf) der letzte Befehl ist. Anderenfalls läuft der Kommandozeilenpuffer über.

Selbstverständlich kann der Einsatz von Makros, Scripts usw. auch auf die Spitze getrieben werden. So ist es möglich innerhalb eines Alias als Befehl den Namen eines ARUNZ-Scripts anzugeben. Der Kommandoprozessor fügt die entsprechende Kommandozeile des Scripts in die des Alias ein und arbeitet dann alles Schritt für Schritt ab. Eine derartige Verknüpfung birgt aber mehrere Risiken in sich.

Schon die kleinsten Veränderungen des Systems, z. B. ein anderer Dateiname oder ein umbenanntes Verzeichnis, können einen immensen Arbeitsaufwand nach sich ziehen. Das ist gerade in einer dynamischen Umgebung wie dem Z-System von großer Bedeutung. Außerdem kann es durch die Auflösung der Platzhalter im Script durch ARUNZ und das anschließende Einfügen in den Alias sehr schnell zum Überlauf des Kommandozeilenpuffers kommen. Man halte sich vor Augen, daß die nur zwei Zeichen lange Angabe „$1“ für das erste Token nach der Auflösung durchaus sechzehn Zeichen lang sein kann (z. B. A15:BEISPIEL.TXT).

Größe der Systemelemente und Substitution durch Programme

Die Flexibilität des Z-Systems schafft es immer wieder, selbst langjährige Anwender zu begeistern. Natürlich wünscht man sich den größtmöglichen Komfort und installiert demzufolge leistungsfähige Systemelemente. Selbstverständlich benötigen diese auch Speicherplatz. Ein Maximal-System bietet zwar allen erdenklichen Komfort, jedoch ist der verbleibende TPA so klein, daß man eigentlich nicht mehr damit arbeiten kann. Es gilt also ein ausgewogenes Verhältnis zwischen Größe und Komfort zu finden.

Inzwischen sind zahlreiche Varianten der Systemelemente verfügbar. Zum Teil steht sogar der Quelltext zur Verfügung und man kann ihn durch einfache Ja-/Nein-Entscheidung anpassen bzw. festlegen, welche Befehle und Funktionen enthalten sein sollen. Es ist zweckmäßig, vor Installation einer bestimmten Variante genau zu überlegen, was unbedingt benötigt wird. Auch eine Überprüfung des Systems von Zeit zu Zeit ist empfehlenswert. Hat man im Laufe der Zeit einen eigenen Arbeitsstil entwickelt, findet man garantiert ein paar überflüssige Funktionen.

Doch man muß sich nicht nur auf Teile von Systemelementen beschränken, sondern kann im Einzelfall das eine oder andere Modul gänzlich verzichten. Bestes Beispiel hierfür ist das IOP (Input/Output Package). Es beinhaltet entweder sehr systemspezifische Routinen für einen bestimmten Computertyp wie z. B. einen Tastaturtreiber oder bietet selten benötigte Funktionen wie z. B. Umleitung der Druckausgabe in eine Datei.

Im Kapitel 6 des Handbuches ist der Aufbau der Z-System-Umgebung beschrieben. Standardmäßig werden neben CPR und DOS folgende Elemente installiert: Warmboot-Umleitung, Environment Descriptor, TCAP, Message Puffer, Pfad, Wheel Byte, externer FCB, Kommandozeilenpuffer, externer Kommandoprozessor-Stack, Shell Stack, optionaler Nutzerspeicher, NDR, RCP, FCP und IOP. Mittels MKZCM kann auf alle genannten Elemente ab Shell Stack (dieser eingeschlossen) Einfluß genommen werden. Gibt man für die Größe „0“ ein, so wird das jeweilige Element nicht installiert.

Nun sollte man keinesfalls das System einer Radikalkur unterziehen. Auch wenn man im ersten Augenblick zu der Überzeugung gelangt ist, man könne auf ein Systemelement verzichten, sollte man sich überlegen, ob eine Minimalvariante vielleicht doch noch Platz hätte. Ein gutes Beispiel hierfür ist der Shell Stack. Sehr schnell erliegt man der Versuchung, ihn mit dem Hintergedanken „brauch' ich eh' nicht“ aus dem System zu entfernen. Doch spätestens beim Versuch, ZFILER zu starten, besinnt man sich eines besseren.

Beim RCP ist die Auswahl verfügbarer Varianten groß und dementsprechend ist der Leistungsumfang sehr verschieden. Neben den standardmäßigen Paketen gibt es inzwischen auch Spezialversionen für den KC. Schon nach relativ kurzer Zeit stellt man fest, daß einige Befehle (z. B. REG, PORT, WHL) nie benutzt werden. Dort sollte die Aufräumungsarbeit beginnen. Die Einsparungen können hier recht üppig ausfallen – von 9 oder 10 Records auf minimale 4 Records. Werden einige Kommandos doch hin und wieder benötigt, muß man deshalb nicht gleich eine neue Systemumgebung laden. Die Funktionen von vielen „internen“ Befehlen lassen sich durch ein entsprechendes Programm nachbilden – etliche sogar als Typ-3 oder Typ-4.

Ähnliches gilt für den FCP. Lediglich die standardmäßig installierte Variante ist nicht ganz so groß, daß man gegenüber der Minimalversion so gravierende Einsparungen machen könnte wie beim RCP. Doch nichtsdestotrotz lohnt es sich auch hier, auf einige Befehle und somit auf ein paar Bytes zu verzichten. Das absolute Minimum stellen die vier Befehle IF, ELSE, FI und ZIF dar. Diese genügen für die meisten Anwen-

dungsfälle und sollten deshalb resident im Speicher verbleiben, um einen schnellen Zugriff zu gewährleisten. Alle anderen Funktionen (z. B. Test auf Vorhandensein einer Datei mit EXIST, Zugriff auf Userregister) können dem Programm IF.COM überlassen werden. Es muß sich nur im Root-Verzeichnis befinden und schon wälzt der Kommandoprozessor alle Befehle zur Fluß Steuerung, die intern nicht vorhanden sind, auf dieses Programm ab.

Völlige Freiheit besitzt man bei der Definition der „benannten Verzeichnisse“ (Named Directories, NDR). Weder das System noch irgendwelche Programme sind darauf angewiesen, weshalb man auch ganz darauf verzichten kann. Die Namen erscheinen an der Eingabeaufforderung und erleichtern so die Orientierung im Dschungel der Verzeichnisse. Außerdem können sie, wie bereits in meiner Artikelreihe geschildert, in Scripts und Makros benutzt werden. Während der Konfiguration mit MKZCM wird lediglich die Anzahl der verfügbaren Namen festgelegt. In einem Record haben jeweils sieben Namen Platz. Das klingt zunächst nicht sehr speicherhungrig, aber so mancher fühlt sich erst mit dreißig oder vierzig benannten Verzeichnissen so richtig wohl. Hier gilt also: die Masse macht's.

Lange Zeit konnte der optional verfügbare Nutzerspeicher (User Memory Area, UMA) vernachlässigt werden. Erst mit ZSDOS/ZDDOS bekommt er durch die Möglichkeit, dort die Treiber für Datums- und Uhrzeitstempel abzulegen, eine Bedeutung. Selbstverständlich gilt auch hier, sich auf die notwendigsten Routinen zu beschränken. Bei den mitgelieferten und verfügbaren Systemelementen ist stets der Platzbedarf im System angegeben, nicht so beim UMA. Hier muß der Anwender selbst feststellen, wieviele Records der gelinkte (!) Treiber benötigt. Grundsätzlich gilt hier, daß zuviel nicht schaden kann. Doch was nutzt es, wenn man bei RCP und FCP spart, während hier die Bytes mit vollen Händen „rausgeschmissen“ werden.

Programmtypen – der kleine Unterschied

Im Handbuch wird fast beiläufig erwähnt, daß es außer der „normalen“ Art von Programmen auch noch welche vom Typ-1, Typ-2, Typ-3 und Typ-4 gibt. Allerdings werden keinerlei Hintergrundinformationen zu diesen speziellen Programmtypen geliefert, dabei haben sie doch einige interessante Eigenschaften. Für den Anwender bleibt beim Aufruf einer COM-Datei der Programmtyp verborgen, es sei denn das Programm selbst weist an irgendeiner Stelle darauf hin, wie z. B. ZPATCH.

Programme vom Typ-3 oder Typ-4 werden zur Ausführung nicht auf die Adresse 0100 hex geladen, sondern in höhere Speicherbereiche. Dadurch bleibt im Normalfall das zuvor geladene Programm (auf 0100h) im Speicher erhalten und kann mittels GO-Befehl wieder gestartet werden, ohne daß es zuvor vom Datenträger geladen werden muß.

Die 5 Programmtypen sollen nun eingehender betrachtet werden. Zunächst einmal die reinen CP/M Programme, die selbstverständlich auch unter dem Z-System laufen. Da sie von der neuen Systemumgebung keine „Ahnung“ haben, benutzen sie sie auch nicht. Normale CP/M Programme werden in den Arbeitsspeicher ab Adresse 0100h geladen. Bis auf wenige (systemnah programmierte) Ausnahmen bereiten sie keine Probleme.

Das Z-spezifische Pendant zu den normalen CP/M Programmen sind jene vom Typ-1. Bis auf einen wenige Bytes großen Header, der dem eigentlichen Programmcode vorangestellt wird, unterscheiden sie sich auf den ersten Blick nicht von standardmäßigen CP/M Programmen. Man kann sie auch unter einem solchen System starten, erhält aber meist eine Fehlermeldung und das Programm wird abgebrochen.

Dies hat natürlich einen Sinn. Im Verlaufe der Programmausführung wird von den speziellen Eigenschaften des Z-Systems Gebrauch gemacht. Weil dies unter einem standardmäßigen CP/M System unweigerlich zu Problemen führen würde, blockt das Programm die Benutzung in einem solchen Fall ab.

Da für alle Zugriffe auf die Eigenschaften des Z-Systems die Adresse des jeweiligen Elementes (z. B. TCAP für Terminalsteuerzeichen) aus dem Environment Deskriptor benötigt wird, muß dieser zuerst gesucht werden. Dazu kann sich der Programmierer einer Routine aus der Z3LIB bedienen. Für die anschließende Initialisierung steht ebenfalls eine Routine zur Verfügung. Diese setzt eine globale Variable auf das Environment und ermöglicht so den anderen Z3LIB-Routinen einen schnellen und unkomplizierten Zugriff auf das Z-System.

Leider konnte ich kein Programm vom Typ-2 auftreiben, der im Handbuch ebenfalls erwähnt wird. Offenbar handelt es sich dabei um eine ausgestorbene Art, über deren Besonderheiten sich nur spekulieren läßt.

Hingegen trifft man gelegentlich noch auf Programme vom Typ-3. Diese besitzen einen ähnlichen Header wie die Typ-1 Programme, mit dem sie sich gegenüber dem Kommandoprozessor identifizieren. Im Gegensatz zu diesen hat der Programmierer als Basisadresse jedoch eine andere als 0100h festgelegt (meist 8000h oder höher). Diese Basis- oder Startadresse ist im Header enthalten und wird vom CPR zum Laden und Starten des Programmes benutzt. Sinn und Zweck von Typ-3 Programmen ist es, den unteren Speicher zur Wiederausführung eines geladenen Programmes „freizuhalten“.

Bei Programmen vom Typ-3 kann jedoch folgendes Problem auftreten: Der Programmierer hat, ausgehend von seinem System, eine sehr hohe Basisadresse festgelegt. Beim Start auf einem System mit weniger TPA

kann es dazu führen, daß Teile des Systems überschrieben würden. Um einem Absturz vorzubeugen, lädt der CPR in diesem Fall das Programm nicht – es kann also nicht ausgeführt werden.

Dieses Problem wird von den Typ-4 Programmen elegant gelöst. Bei ihnen wird die Basisadresse nicht vom Programmierer festgelegt, sondern erst zur Laufzeit bestimmt. Dazu sind jedoch zwei besondere Voraussetzungen notwendig. Der Programmcode darf nicht auf eine feste Adresse gelinkt werden, sondern muß als PRL- oder SPR-Datei erzeugt werden. In die ersten beiden Records, die bei PRL-/SPR-Dateien standardmäßig nur die Länge des Codes enthalten, wird ein spezieller Header eingebettet. Darin sind Routinen zur Berechnung der Ladeadresse und zum Linken des Programmcodes enthalten.

Wird vom Kommandoprozessor der Typ-4 erkannt, so lädt er den ersten Record des Programms in den standardmäßigen DMA-Puffer (Adresse 0080 h). Anschließend wird ein CALL dorthin ausgeführt, um die Ladeadresse berechnen zu lassen. Danach lädt der CPR den zweiten Record in den DMA-Puffer und die restliche Datei (also den Programmcode incl. Bitstream) ab berechneter Ladeadresse in den TPA. Ist dies geschehen, wird durch einen weiteren CALL die Routine im DMA-Puffer aufgerufen. Diese linkt den Programmcode und kehrt anschließend zum Kommandoprozessor zurück, der dann das Programm startet.

Auf diese Weise werden Programme vom Typ-4 stets auf die höchste verfügbare Adresse geladen, wobei der untere TPA-Bereich unverändert bleibt. Ein vorher geladenes Programm kann einfach mittels GO Befehl wieder gestartet werden. Ein Überlauf des TPA kann während des Ladevorgangs nicht auftreten.

An dieser Stelle möchte ich alle Programmierer ermutigen, es einmal mit Typ-4 zu versuchen. Grundsätzlich kann jedes Programm dazu eingerichtet werden. Für den speziellen Typ-4-Lader stehen Quelltexte (auch in deutsch) und ein einfach zu handhabendes Overlay zur Verfügung. Mit Hilfe des Original LINK von Digital Research ist es möglich, anstelle einer COM-Datei das PRL- oder SPR-Format auszugeben. Es mangelt also nicht an Werkzeugen!