RUHR - UNIVERSITÄT BOCHUM

Arbeitsberichte

des

Rechenzentrums

Direktor: Prof. Dr. H. Ehlich

Nr. 7209

Anregungen und Kritik zum Betriebs- und Programmiersystem der TR 440

von
Prof. Dr. H. Ehlich

Bochum, September 1972

Inhaltsverzeichnis

Vorwort

Der vorliegende Bericht ist aus Erfahrungen mit dem Betriebssystem BS3 der TR 440 entstanden. In ihm sind einige Punkte herausgegriffen, die für die Benutzung der Anlage von Bedeutung sind. Sie betreffen natürlich nur solche Fälle, wo mir eine Verbesserung dringend erscheint.

Das Ziel ist es, eine intensive Diskussion zu eröffnen. Daher setzen die Ausführungen die Beherrschung des BS3 voraus. Viele Fragen werden nur angeschnitten. Eine Klärung bedarf weiterer Untersuchungen.

I. Kommandosprache

1. Grundsätzliches

Die Kommandosprache dient zum bequemen Umgang des Benutzers mit der Anlage und gestattet es, Dienstleistungen des Systems leicht zu erhalten. Da die Kommandosprache bei großen Systemen sehr umfangreich sein kann, muß sie leicht erlernbar und daher logisch aufgebaut sein. Auf der anderen Seite soll sie so gestaltet werden, daß der Programmaufwand (Speicher und Zeit) minimiert werden kann. Die jetzt für den TR 440 vorliegende Kommandosprache erfüllt einige dieser Forderungen hervorragend (z.B. Abkürzungen), andere wiederum gar nicht (vgl. z.B. Synonyme: QUELLE, INFORMATION, TDATEI oder die verschiedene Reihenfolge ZIEL-INFORMATION bzw. INFORMATION-ZIEL).

Im Laufe der Zeit hat sich außerdem herausgestellt, daß mehrere Angaben in einem Spezifikationswert durchgeführt werden, weil sie logisch zusammengehören. Als Trennzeichen werden dabei aber sowohl "Punkt" wie "Bindestrich" (Minuszeichen) benutzt. Siehe etwa

DATEIBASISNAME.DATEINAME
nat.Zahl - Datenbasisname.Dateiname
RAM-P
MB(EXDKZ)m.n.

2. Notwendige Erweiterung der Kommandosprache

Die obigen Beispiele zeigen, daß eine Erweiterung der Kommandosprache unumgänglich ist, wenn nicht die einzelnen Kommandos durch zu viele Spezifikationen unübersichtlich lang werden. Dafür sollte in der Sprache ein einheitliches Sprachmittel geschaffen werden. Der folgende Vorschlag läßt sich im Rahmen der vorliegenden Kommandosprache wohl am leichtesten logisch einführen. Jede Spezifikation wird unterteilt in Unterspezifikationen, die der Übersichtlichkeit halber aber keinen Namen erhalten, aber bei Abarbeitung von links nach rechts numeriert werden. Jeder dieser Unterspezifikationen wird ein Unterspezifikationswert zugewiesen. Unterspezifikationswerte werden voneinander durch das "Minuszeichen" getrennt. Die Syntax der Kommandosprache ist dann im Punkt 3.1 auf Seite 11 des Kommandohandbuchs folgendermaßen zu ändern:

3.1.a. < Spezifikationswert > ::=
< Unterspezifikationswert > [- < Unterspezifikationswert >]$^{\infty}$
[' < Unterspezifikationswert > [- < Unterspezifikationswert >]$^{\infty}$]$^{\infty}$

3.1.b. < Unterspezifikationswert > ::=
< Teilwert > ['< Teilwert >]$^{\infty}$ | < Symbol für "undefiniert" >

Ferner ist dann 3.8. auf Seite 12 des Kommandohandbuchs zu ändern:

3.8. ‹Symbol für "undefiniert"› ::= &

sowie 3.10. auf Seite 12 im Kommandohandbuch:

3.10. ‹Zeichenkette von Typ 1› ::=
‹Zeichenfolge, in der die Zeichen
◊|*| /| ,| -| '| = |(|) nicht auftreten

Und ebenso ist die Fußnote 4) auf Seite 12 zu ändern:

4) Die nur aus dem Zeichen "&" bestehende Zeichenkette der Länge 1 wird nicht allgemein als Normalstring, sondern speziell als Symbol für "undefiniert" verstanden (vgl. 3.8.).

Die Benutzung des Minuszeichens als Sprachmittel für die Abgrenzung von Unterspezifikationswerten hat die notwendige Folge, daß für "undefiniert" ein anderes kurzes Symbol benutzt werden muß. Der obige Vorschlag, etwa "und" zu nehmen, ist ein möglicher unter vielen anderen. Es könnte z.B. der Buchstabe "u" gewählt werden.

Dadurch, daß Unterspezifikationen keinen Namen erhalten, muß eine Regelung getroffen werden, wie eine Adressierung vorgenommen wird. Hier bietet sich das Zählen der aufgetretenen Minuszeichen an. Z.B. könnte die Wertzuweisung für die 1. und 3. Unterspezifikation mit folgender Angabe geschehen:

< Normalstring > -- <Normalstring>

oder für die 3. Spezifikation alleine durch die Angabe:

-- < Normalstring > .

Selbstverständliche Forderung ist, daß jede Unterspezifikation vorbesetzt werden kann. Neben der logischen Forderung, eine übersichtliche Programmsprache zu erhalten, ist dies mit ein Grund für die Einführung der Unterspezifikationen.

Im Verlauf dieses Bereichs wird im folgenden stets davon ausgegangen, daß der obige Vorschlag in dieser Weise realisiert ist. Bei den Ausführungen über Datenmanipulationen wird in Beispielen die Nützlichkeit sichtbar.

3. Zur Implementierung

Hier müssen sich einige Worte zum Entschlüssler anschließen. Grundsätzlich sollte der Entschlüssler kein Kommando selbst verarbeiten, wie es jetzt geschieht. (Das jetzige Vorgehen ist wahrscheinlich nur historisch bedingt.) Einzige Ausnahme bilden wohl die Kommandos, die den Entschlüsslerlauf beeinflussen (z.B. ESDUMP,..). Alle anderen Kommandos werden in Operatoren ausgeführt. Nur so erreicht man eine leichtere Maintenance und Änderung.

Ferner sollen alle Kommandonamen, Spezifikationsnamen und Spezifikationen, einschließlich der Tabellen, im Entschlüsslergedächtnis liegen. Die Angabe von TAB(i) bezieht sich dann auf eine schon im Gedächtnis liegende Tabelle. Es muß aber im DEFINIERE-Kommando die Möglichkeit geben, eine neue Tabelle zu definieren. Etwa durch die Angabe

(TAB(i) = A-1'KØ-2'BIN-4)

War TAB(i) schon bekannt, so Fehlermeldung falls nicht identisch, sonst Eintragen. Bei Löschung eines Kommandos wird eine evtl. zugehörige Tabelle nur gelöscht, wenn sie in keinem anderen Kommando benutzt wird.

Grundsätzlich sollten alle Kommandos über das DEFINIERE Kommando erstellt werden und daher ganz einheitlichen Aufbau nach der Syntax haben. Das Definiere-Kommando muß entsprechend dem obigen Vorschlag erweitert werden, und zwar (mit den bisherigen Bezeichnungen) muß in der Liste der möglichen Unterspezifikationswerte das Minuszeichen zulässig sein, etwa

...,SPEZIFIKATION=DATEI(NZ-DT-N,N)'...

Im übrigen sei noch auf folgendes hingewiesen. Die starre Reihenfolge obligat, optional behindert sehr den logischen Aufbau der Kommandos. Nur aus diesem Grund sind oft Umstellungen in der Reihenfolge notwendig. Es sollte bei jeder Spezifikation vermerkt werden, ob sie obligat ist oder nicht. Für das DEFINIERE-Kommando bedeutet dies die Einführung eines Typs "ØB", der etwa in der Auflistung unter NL stehen könnte. Ferner ist für viele Zwecke die getrichelte Linie (Seite 7 Kommandohandbuch) zu hoch gezogen, sie gehörte unter die Spezifikation N, damit, falls etwa

(NZ,N)

angegeben ist, im Fall daß NZ nicht vorliegt, dem Operater ein Normalstring übergeben werden kann.
Zur Zeit macht sich unangenehm bemerkbar, daß für TRAEGER keine Spezifikation TR besteht. In vielen Operatoren muß

nun bei Einführung neuer Traeger (UMB, WSP, W14) sehr viel geändert werden. Daher sollte für das Definiere-Kommando ein Typ Traeger (TR) unbedingt eingeführt werden.

Der Vorschlag, Unterspezifikationen einzuführen, bedingt, daß es zwei Arten von Listen gibt, die gemischt auftreten können. Für die Liste der Unterspezifikationen müßte dann im Startsatz ein neuer Parametertyp eingeführt werden, der etwa durch eine 8 gekennzeichnet ist.

Die Einführung des obigen Vorschlages bringt aber für die Entschlüsselung nicht nur größeren Aufwand , sondern auch Erleichterungen, einzelne Typen wie etwa PZ, NDT können wegfallen. Bei Typ GER und GDT hängt dies von der Ablage ab. Grundsätzlich ist in obigem Konzept

GER durch QN-QN-QN

zu ersetzen.

Im Verlauf der Ausführungen wurde mehrfach auf logischen Aufbau und leichte Handhabung (Erlernbarkeit, nicht dauernd nachschlagen) hingewiesen. Darauf sollte vor allem auch bei den Spezifikationsnamen und der Reihenfolge geachtet werden. Bei fast allen Kommandos werden Daten transportiert oder verarbeitet. Dies läßt sich vielleicht

am besten durch das Fragepaar

woher ? - wohin ?

umschreiben. Für die entsprechenden Spezifikationen sollten einheitliche Namen gefunden werden, etwa

Quelle Ziel

oder

Information Ziel

und diese Reihenfolge eingehalten werden.

Für Manipulationen am Entschlüsslergedächtnis sollte ein besonderer Operator zuständig sein, der die beiden Kommandos ◇DEFIN. und ◇GED. kennt.
Dabei sollte das letztere durch
◇DEFIN., GEDAECHTNIS, PS&GEDAE., ZIEL (STD,DT/GER,NL,OB)
gekennzeichnet sein. Die Ausgabe sollte nicht binär sondern als Klartext des Definiere-Kommandos so erfolgen, daß die Datei bzw. Karten direkt vom Definiere-Kommando verstanden werden, also mit Fluchtsymbol usw.. Das Definiere-Kommando muß natürlich stets im Entschlüsslergedächtnis sein, alle anderen können gelöscht werden.
Die Angabe -STD- unter Ziel sollte eine Normierung des Gedächtnisses (zusammenschieben, Bereinigen von Lücken,

usw.) bewirken, so daß jeder Benutzer seinen Speicherplatz reduzieren kann.

II. Datenhaltung - Datensicherung

1. Grundsätzliches

Bei der Konzeption der TR 440 war man sich der Probleme der Datenhaltung, z.B. der Menge der anfallenden Daten, insbesondere auch für kommerzielle Aufgaben, nicht bewußt. Bei der Bearbeitung der formalen Sprachen wurde die große Erfahrung von der TR 4 ausgenutzt. So entstanden für die Sprachen, so verschieden sie auch sein mögen, einheitliche Kommandos. Bei der Datenhaltung herrscht hingegen eine große Unübersichtlichkeit. Diese reicht bis in die logische Struktur. Darum wird hier ein konsequenter Aufbau für die Datenhaltung vorgeschlagen. Dabei steht der Gesichtspunkt des Anwenders im Vordergrund; an zweiter Stelle muß die Hardware berücksichtigt werden, und erst zum Schluß kommen Fragen der Implementierung.

2. Aufbau der Daten von Benutzerseite

Für den Benutzer ist seine Datenmenge strukturiert. Und zwar gliedert sie sich logisch nach Sachgesichtspunkten Programme, Personaldaten, Haushaltsdaten, Gebäudedaten, Produktionsdaten, Vertriebsdaten usw..
Jede der hier aufgezählten Gruppen untergliedert sich

weiter, etwa Personaldaten in Studenten, Verwaltungspersonal, Hochschullehrer oder Programme in die einzelnen Programme.

Um zu einer einheitlichen Darstellung zu kommen, wird daher folgende Sprachregelung vorgenommen:

Der Anwender hat seine Daten in DATEIEN (file). Dateien sind eine Zusammenfassung von Daten unter einem Gesichtspunkt. Mehrere DATEIEN faßt der Benutzer zu einer BIBLIOTHEK zusammen. Die Gesamtheit seiner BIBLIOTHEKEN bildet den Datenbestand des Benutzers. Um Verwechslungen zu vermeiden, ist eine DATEI durch den DATEINAMEN und den BIBLIOTHEKSNAMEN eindeutig gekennzeichnet. (Dies ist übrigens genau das Konzept Datenbasis-Datei).

Jede DATEI unterteilt sich in Sätze (records). Jeder Satz enthält eine gewisse Anzahl Elemente (Zeichen, Ausgabezeichen, binäre Worte).

Ferner ist bezüglich des Zugriffs jede Datei einheitlich aufgebaut. Bei Zugriff ist nicht nur Sequentiell und Random wichtig, sondern auch die Zugriffsberechtigung.

3. Die Hardwarerealisierung

Zur Aufbewahrung von Daten - nach den obigen Dateien und Bibliotheken - hat der Benutzer und intern die Datenverarbeitungsanlage große Möglichkeiten. Es existieren z.Z. mannigfache externe Speicher, Lochkarten, Lochstreifen, Magnetbänder, Magnetplatten, Magnetkarten, usw.. Daneben existieren bei größeren Rechnern interne Speichermedien, solche die während der Bearbeitung zur Verfügung stehen und solche, die intern dauernd zur Verfügung sind (Arbeitsspeicherraum und langfristige Datenhaltung (LFD)).

Jedes dieser Speichermedien bezeichnen wir als SPEICHERKLASSE, die durch eindeutige, möglichst kurze Namen bezeichnet werden. Solche sind MB, WP, LF, T, P usw..
Die einzelnen Klassen unterteilen sich nun weiter in einzelne TRAEGER (Bänder, Türme, Benutzer bei LF). Der einzelne Träger wird durch seinen NAMEN eindeutig gekennzeichnet.

4. Folgerungen

Eine Datei ist als solche also eindeutig gekennzeichnet durch vier Angaben:

BIBLIOTHEKSNAME, DATEINAME,
SPEICHERKLASSENNAME, TRAEGERNAME.

Da die beiden ersten von der logischen Struktur, die beiden letzten durch Hardwareaufteilung entstehen, ist eine sinnvolle Darstellung

DBNAME.DTNAME - SPNAME.KNAME .

Der Dateiname sollte wie bisher bis zu 12 Alphazeichen und in Klammern Generationsnummer und Versionsnummer enthalten. Alle anderen Namen sollten höchstens 6 Alphazeichen lang sein. An den Trägernamen kann sich in Klammern die Dateiabschnittsnummer und die Dateifolgenummer anschließen. Zur Illustration folgendes Beispiel:

ANTØN.FRIEDA(3.7) - MB.3OOO11(2.1O)

Folgende implizit-Regelung wäre weiter wünschenswert:

Fehlt im ersten Teil der Punkt, so ist eine Standardbibliothek gemeint. Wird dagegen nur ein Name mit abschließendem Punkt angegeben, so sind alle Dateien der Bibliothek adressiert. Wird dagegen nur eine Speicherklasse und evtl. Träger angegeben, so sind alle

Bibliotheken mit ihren Dateien auf dieser Speicherklasse (wichtig bei Trommel und Platte) oder diesem Träger gemeint.

5. Typ und Art einer Datei

Die Typen SEQ, RAN, RAM und PHYS sind wohl unentbehrlich. Obwohl vom Benutzer her oft RAN als überflüssig erscheint, ist aus Speichergründen an diesem Typ festzuhalten. Auch die Angaben U,M,G haben sich bewährt, vor allem bei der LFD.

Anders sieht es dagegen bei den Angaben zum Satzbau O,W,A usw. aus. Diese Arten des Satzbaues sind wohl nötig wegen der Ausgabe (Drucker, Stanzer). Alle anderen sind dagegen überflüssig. Ferner ist die bisherige Behandlung für den Benutzer eine Zwangsbeglückung. Für den Inhalt der Benutzerdateien ist allein der Benutzer, nicht das System zuständig. Alle Kommandos zur Dateimanipulation müssen auf jeder Position Dateien mit jedem Satzbau zulassen mit folgender Implizitregelung: Bei Transport von Ø,W $\longrightarrow$ A wird ein Steuerzeichen (NL) vorne eingesetzt. Bei Transport von A $\longrightarrow$ Ø,W wird das Steuerzeichen unterdrückt.

6. Zur Datensicherung

Wie bisher ist ein Passwort wohl unumgänglich. Dies ist aber nicht ausreichend. Auch die Angabe zum Koordi - nationstyp ist hier zu global. Bei den Aufgaben, die auf die Datenhaltung zukommen, wird der Gesetzgeber Auflagen erlassen (Datenschutzgesetz). Der Zugriff zu Dateien muß daher neu überdacht werden. Ein Vorschlag wäre der folgende: Bei Kreation einer Datei wird eine Liste mitgegeben, in der Zugriffsberechtigungen angegeben werden. In der jetzigen Terminologie etwa

BKZ.FKZ - L(a-e)

oder

BKZ.FKZ - S(a-e)

BKZ = Benutzerkennzeichen,
FKZ = freies Kennzeichen,
L = lesen,
S = schreiben,
(a-e) Zeitintervall.

Fehlen diese Angaben, so kann nur unter dem bei Kreation gültigen BKZ.FKZ zugegriffen werden. Außerdem ist eine Spezifikation ALL für BKZ.FKZ vorzusehen. Um Gruppen von Benutzern zu ermöglichen, muß auch die Angabe von FKZ ausreichen (Regelung wie bei Datei). Außerdem ist

BKZ mit mindestens 12 Alphazeichen vorzusehen. Es werden jedoch nur die in der Liste angegebenen Zeichen geprüft. Steht in der Liste dann etwa

700001EH.GEH - s(0-24)

so kann auch mit

BKZ.FKZ = 700001EHLICH.GEHEIM

zugegriffen werden. Dateikenndaten und Passwort können nur von einem Schreibberechtigten, wenn kein anderer Zugriff vorliegt, geändert werden.

Zur Koordinierung ist folgendes zu sagen: Die bisherige Regelung ist unbefriedigend. Grundsätzlich sollten alle Dateien der LFD und Wechselplatte koordiniert werden. Aber die Koordinierung muß auf SATZEBENE NICHT DATEIEBENE geschehen. Nur so ist bei vielen Aufgaben überhaupt sinnvolles Arbeiten möglich. Das dies keine besonderen Schwierigkeiten macht, haben mehrere Diskussionen in Konstanz ergeben. Gerade die hier angeschnittenen Fragen sind überaus wichtig im kommerziellen und administrativen Bereich. Eine gründliche Diskussion aller Beteiligten erscheint unerläßlich.

7. Kommandos für Dateimanipulationen

Die jetzige Kommandosprache zeichnet sich hier durch eine überaus große, unübersichtliche Fülle aus. Viele dieser Kommandos erledigen nur Spezialfälle. Der Benutzer muß sich jeweils überlegen, welches er nun nehmen soll. Der unter II.4. gemachte Vorschlag der Dateikennzeichnung würde es erlauben, die Dienstleistungen vieler Kommandos durch das folgende zu ersetzen:

```
↵DEFIN.,KØPIERE,PS&DAT,QUELLE(DT-TR,GER,F)'SATZ(QN-QN)'
  ZIEL(DT-TR,GER)'PRØTØKØLL(TAB(i))'NUMERIERUNG(NUM-QN)'
  ART(QN).
```

Trifft man die implizit-Regelung, daß RAM und RAN Dateien auf MB zuerst die Satznummer enthalten, könnten die Kommandos Eintrage, Stanze, Drucke, Zeichne, Teintrage, Tkopiere, Verlagere, Sichere, entfallen.

Das oben genannte Kommando leistet aber mehr als das bisherige, denn es können ganze Bibliotheken oder auch ganze Träger bearbeitet werden.

Für die bequeme Korrektur von Sätzen einer Datei im Gesprächsmodus ist das Kommando TZKØR. nicht befriedigend. Das Kommando müßte erweitert werden, damit z.B. auch der 1. Buchstabe einer Zeile durch einen Zwischenraum (blank)

ersetzt werden kann oder auch Einfügungen und Löschungen gemacht werden können. Dazu folgender Vorschlag:

Es werden drei Zeichen Z1, Z2, Z3 eingeführt. Alle Zeichen des alten Satzes, unter die Z1 geschrieben wird, werden übernommen (wie bisher blank am Anfang). Alle Zeichen, die nach Z2 stehen, werden in den Satz eingefügt, und zwar vor das Zeichen, unter dem Z2 steht (vor, damit auch am Satzanfang Einfügung möglich wird). Alle Zeichen des alten Satzes, unter denen Z3 steht, werden gelöscht.

Da Z1, Z2, Z3 Zeichen des normalen Zeichensatzes sind, ist eine schnelle Änderung notwendig. Daher folgende Regelung: Nach Angabe der Satznummer kann durch Komma getrennt die Angabe Z1-Z2-Z3 erfolgen.

Etwa

GIB ZEILENNUMMER ◇ : 1134, !-y-Z ◇ .

8. Darstellung der Daten im Rechner

Aus historischen Gründen gibt es bei der TR 440 das Gebiets- und Dateikonzept. Für den Benutzer ist eigentlich nur die Dateilage wichtig. Das Gebietskonzept hat aber zur Folge, daß zumindest die Eingabedaten unnötig oft zeichenweise

verarbeitet werden müssen. Hier wird viel Rechnerzeit verschwendet. Deshalb sollte eine Ablage gefunden werden, die das unnötige Durchsuchen auf Satzende überflüssig macht. Schon die Eingabe-Vermittler könnten die Daten so ablegen, daß am Anfang (und Ende) des Satzes seine Länge steht. Dadurch würde erreicht, daß die bei der Eingabe bekannten Informationen nicht weggeworfen und später wieder mühsam gewonnen werden müssen.

Wie die Daten intern transportiert und verwaltet werden (Satz- oder Kacheltransporte) ist eine Sache des Betriebssystems. Der wichtigste Gesichtspunkt ist dabei: beste Ausnutzung der Hardware.

III. Dialogbetrieb

1. Vorrangssteuerung für Gespräche

Die bisherige Regelung,zwischen Kurz- und Langläufen im Gespräch zu unterscheiden, ist nicht befriedigend. Vor allem bei Kommandos, die nur minimale Dienstleistungen fordern, wird die Reaktionszeit unerträglich lang. Dabei wird unter Reaktionszeit die Zeit verstanden, die zwischen dem Eintippen des abschließenden Codes der Eingabe und dem Abdruck des 1. Zeichens der Ausgabe verflossen ist. Bei gleichzeitigem Betrieb von 40 Konsolen liegt diese Reaktionszeit nach unseren Messungen in der Größenordnung von 10 und mehr Sekunden. Dies ist bei dem Korrigieren von Zeichen einer Zeile unerträglich lang. Deshalb sollte folgende Regelung getroffen werden:

a) Bestimmte Operatoren des Betriebssystems werden bei Eingabe bestimmter Kommandos als Sonderkurzläufe gekennzeichnet. Die arbeitswilligen Sonderkurzläufer werden stets vor allen übrigen abgearbeitet.

b) Auch der Benutzer kann sein Objektprogramm als Sonderkurzlauf definieren. Um Mißbrauch zu verhindern, wird eine obere Zeitschranke für Sonderkurzläufe festgelegt. Überschreitet ein Sonderkurzlauf diese

Zeitschranke, wird er für eine lange Strafzeit überhaupt nicht mehr berücksichtigt.

2. Fehlerbehandlung bei Entschlüssler und Operatoren

Die Behandlung von Fehlern bei der Eingabe von Kommandos in Gesprächen muß dringend verbessert werden. Fehlen z.B. obligate Spezifikationswerte, so sollte der Entschlüssler diese an der Konsole anfordern und dann weiterarbeiten. Ebenso bei der Angabe falscher Spezifikationswerte. Diese Forderungen können mit dem Wort GESPRAECHSFAEHIGER ENTSCHLUESSLER beschrieben werden. Auch alle Standardoperatoren sollten sich in gleicher Weise verhalten.

IV. Problemorientierte Sprachen

TRACE und DUMP auf Quellenebene haben sich sehr bewährt. Bezugspunkt ist hierbei zunächst die Karten- oder Zeilennumerierung. Dies ist gerade bei TRACE sowohl zeit- wie papieraufwendig und eigentlich auch nicht auf Quellebene, sondern auf Hardwareebene. TRACE auf Quellebene, d.h. mit Angabe der zu überwachenden Variablennamen, wäre sehr viel wirkungsvoller. An dieser Stelle müssen sicherlich neue Überlegungen angestellt werden.

Die Bearbeitung größerer Dateien auf Sprachebene läßt sich zwar in allen Gesprächen durchführen. Es ist aber für den Benutzer unverständlich, wieso auf viele Dateien, die in die Sprachen bearbeitet werden, die Texthaltungskommandos nicht anwendbar sind. Hinzu kommt, daß durch Änderung weniger Zeilen in PS&TEXTHALT mit Ausnahme von TZKOR. (Vergrößerung der Arbeitsspeicher und der Abfragegröße) die Beschränkung auf 160 Oktaden überflüssig wird. Die zu erhaltenten Texte bestehen aber nicht immer nur aus Programmpaketen! An dieser Stelle wäre ein einheitliches Konzept wirklich notwendig.

V. Installationen

Jedes System wird so ausgelegt sein müssen, daß die verschiedenen Installationen unterschiedliche Hardwarekonfigurationen besitzen. Dies ist durch die Finanzkraft und Aufgabenstellung des jeweiligen Anwenders unumgänglich. Aus diesem Grund müßte es an zentraler Stelle des Systems eine installationsspezifische Tabelle geben, in der alle diese unterschiedlichen Merkmale festgehalten sind. Dies wird um so wichtiger, wenn Fremdgeräte, die ursprünglich nicht vorgesehen waren, angeschlossen werden. Auch die EA-Programme der Compiler müßten sich aus dieser Tabelle die gerätespezifischen Angaben holen (z.B. Breite des Druckers, Typensatz des Druckers, Breite des Sichtgerätes usw.). Es ist nicht einzusehen, daß immer wieder Einzelprogramme auf bestimmte Gerätetypen umgestellt werden müssen. Die hier geforderte installationsspezifische Tabelle müßte natürlich für das System einen bestimmten Aufbau haben. Die Änderung oder Neuanlegung einer solchen Tabelle sollte aber auf einer Sprachebene geschehen, die für jeden Benutzer leicht verständlich ist.

VI. Austausch von Daten

Für den Ausbau von Daten zwischen Rechenzentren mit Anlagen unterschiedlicher Hersteller muß ein einheitliches Konzept unbedingt gefunden werden. Dies ist um so wichtiger, als in nächster Zeit umfangreiche Datenbänke aufgebaut werden. Für den Austausch der Daten wird wohl in naher Zukunft immer noch das Magnetband der bevorzugte Datenträger sein. Der Austausch von Plattentürmen ist nach Aussagen der Hardware jedenfalls heute noch sehr problematisch. Dies bedeutet, daß die Dateien auf Magnetband nach international gültigen Normen aufgezeichnet werden und ein entsprechender Code verwandt wird. In dieser Hinsicht ist die Entwicklung des neuen Anpasswerkes für die Magnetbandgeräte und die Hardwareumcodierung ein erster Schritt, wenigstens IBM kompatibel zu werden. Grundsätzlich sollte man sich bei der Bearbeitung der Magnetbänder an die ECMA-Norm halten, in der ja alle Möglichkeiten (auch für verklammerte Dateien) gegeben sind. Ein solches Vorgehen hat den Vorteil, daß wenigstens ein großer Teil der auf anderen Systemen aufgestellten Bänder ohne besondere Aufwendung lesbar ist.

Auf der anderen Seite muß es möglich sein, beliebige Information in beliebiger Blockung binär zu lesen und mit entsprechenden Programmen zu bearbeiten. Es wird sich

nie vermeiden lassen, daß aus irgendwelchen Gründen die Meßtechnik sowohl Bänder wie Lochstreifen in unmöglicher Codedarstellung anliefert. Hier macht sich z.B. äußerst unangenehm bemerkbar, daß der Wagenrücklauf (carriage Return) in SC1 und SC2 überlesen wird; Streifen von automatischen Meßgeräten müssen daher immer eine Sonderbehandlung erfahren, weil sonst ein Satzende nicht erkannt wird.