RUHR-UNIVERSITÄT BOCHUM

ARBEITSBERICHTE

des

Rechenzentrums

Direktor: Prof. Dr. H. Ehlich

Nr. 7604 BO.E6.04

AUFBEREITE

Ein universell einsetzbarer Dateiänderungsoperator für verschiedene Datei- und Dialoggerättypen und/oder Betriebsarten

von

Rainard Buchmann

Juni 1976

4630 Bochum, Universitätsstr. 150, Geb. NA

INHALTSVERZEICHNIS

0. VORWORT

Aus der Kenntnis der inzwischen mannigfaltigen Versuche, für das Programmiersystem des TR440 einen praktikablen Editor zur Verfügung zu stellen, sowie der häufig fehlgeschlagenen Experimente anderer Softwarehersteller wurde das Programm AUFBEREITE unter dem Aspekt geschaffen, die positiven Momente anderer Editoren möglichst zu integrieren und auszuweiten und die negativen zu vermeiden.

Außer den selbstverständlichen Effektivitätsforderungen wurde besonderer Wert auf folgende 3 Gesichtspunkte gelegt:

A) Portabilität

Um das Programm auch auf anderen Anlagen als dem TR440 adaptieren zu können, wurde es in ALGOL60 geschrieben, wobei auf ein wohldefiniertes Paket von E/A-, Stringhandlung- und ähnlichen Schnittstellen-Routinen zurückgegriffen werden konnte ([1], [2], [3]).

B) Ausschöpfung aller Möglichkeiten des TR440-File-Managements.

C) Leichte Erlernbarkeit und Handhabung durch die Konstruktion der Befehls-Syntax und dadurch, daß der Operator sich selbst beschreiben kann.

I. MOTIVATION DER SPEZIELLEN REALISIERUNG

A) Es können alle wichtigen Dateitypen des TR440 bearbeitet werden - also SEQ-, RAN- und RAM-Dateien.

Auf die Behandlung von PHYS-Dateien wurde verzichtet, weil sie aufgrund der fehlenden logischen Struktur zur Aufnahme von Zeichenfolgen ungeeignet sind, und auf die Behandlung von RAS-Dateien, weil diese allgemein vom Programmiersystem nicht unterstützt werden und nur in COBOL bearbeitbar sind.

Innerhalb der SEQ-, RAN- und RAM-Dateien sind alle sinnvollen Satzbauarten erlaubt, also:

1. bei ungefährer Satzlänge alle Elementarten;
2. bei maximaler und genauer Satzlänge die Elementarten:

   Oktaden
   Ausgabezeichen
   Ganzworte

Es sind beliebige Satzlängen und beliebige Satznummern bzw. Satzmarken bearbeitbar, und zwar auf der Basis von einem Ganzwort als Satzmarke, also 48 Bits beim TR440.

B) Zeichen mit dem Zentralcodewert 0 bis 255 sind sämtlich benutzbar - deshalb sind alle Zeichen, die eine Steuerfunktion haben, dynamisch umstellbar.

C) Der Operator ist sowohl im Dialog als auch im Batch benutzbar ; im Batch natürlich mit der Einschränkung, daß einige Möglichkeiten entfallen, die wesentlich auf den Eigenschaften des Dialogs beruhen.

D) Es wurde Wert darauf gelegt, den Benutzer weitestgehend vor versehentlicher Zerstörung seiner Dateien und allgemein vor Informationsverlust zu schützen:

1. Korrekturen werden immer sofort in der Originaldatei durchgeführt, da sonst leicht Korrekturen umsonst gewesen sind, wenn der Systemlauf vor der Sicherung der Korrekturen stirbt.

2. Vor jeder Konsol-Eingabeforderung werden alle Dateibearbeitungen beendet, um der Gefahr vorzubeugen, daß bei einem Systemzusammenbruch Dateien zerstört werden, weil durch nicht ausgestülpte Datenbasispuffer Inkonsistenzen in der Dateistruktur auftreten (Informations- und Stellvertretergebiete passen nicht zusammen).

3. Wird eine Datei durch Eintragen oder Korrigieren zu klein, so versucht der Operator zunächst, sie durch Reservierung zu vergrößern.

   Gelingt dies nicht, gibt er dem Benutzer die Möglichkeit, sie bereinigen zu lassen oder durch TR440-Kommandos Platz zu schaffen, um danach an der Unterbrechungsstelle fortzufahren.

Durch diese Möglichkeiten entfällt der sonst übliche Informationsverlust, wenn aus Platzgründen nur ein Teil der (eingelesenen) Information eingetragen werden konnte.

4. Mit zwei Methoden kann sich der Benutzer selbst davor schützen, daß seine Datei zerstört wird:

   a) Wahlweise werden alle Sätze, die verändert oder überschrieben werden, mit ihrem Inhalt vor der Veränderung in einer "Systemsicherungsdatei" aufbewahrt, um jederzeit alle Änderungen rückgängig machen zu können, die nicht beabsichtigt waren oder sich als ungeschickt erwiesen.

   b) Der Benutzer kann einen Dateiausschnitt festlegen, in dem er arbeiten will. Alle Änderungen und Zugriffe außerhalb dieses "Arbeitsfensters" werden vom Operator abgelehnt.

      Dabei werden als Arbeitserleicherung alle Satznummern mit dem Beginn des "Arbeitsfensters" translatiert, so daß dieses wie eine eigenständige Teil-Datei gehandhabt wird.

      Diese Teil-Dateien sind besonders vorteilhaft in Verbindung mit der in [4] geschilderten Datenhaltung (RB&QUELLHALT).

5. Das oberste Ziel war, allgemein gesagt, dem Benutzer ein komfortables System zum variablen, bequemen Verändern von Dateiinhalten zur Verfügung zu stellen, das den Aufwand an Arbeitszeit, also Konsolsitzungszeit, minimieren helfen sollte. Dabei war es wichtig, den Operator so modular zu gestalten, daß die verschiedenen Komfortstufen die Rechenzeit und den Kernspeicherbedarf nur belasten, wenn sie wirklich benötigt werden.

   Um lauffähig zu sein, benötigt AUFBEREITE 27 K Kernspeicher.

   Aus Gründen der Effektivität nimmt der Operator bis ca. 36 K KSP auf, wenn ihm soviel zur Verfügung steht.

   Bei der Entschlüsselung der Konsoleingabe wurde besonderer Wert auf komfortable Fehlererkennung etc. gelegt, dabei jedoch darauf geachtet, daß in keinem Fall die Rechenzeit eines Bedienungsintervalles überschritten wird, um gute Reaktionszeiten zu erzielen.

   Extreme Rechenzeitoptimierung jedoch wurde lediglich auf Tätigkeiten verwandt, die sowieso lange dauern, etwa assoziative Zugriffe auf die Datei oder Transport von großen Datenmengen, um auch hierbei möglichst schnell Resultate liefern zu können.

6. Der oberste Gesichtspunkt in punkto Handhabbarkeit war, einen möglichst einfachen Zugang zur Syntax der Befehle zu gewähren.

   Deshalb wurde die gleiche Syntax wie die der TR440-Kommandosprache ([5]) gewählt - weil diese erstens leicht erlernbar ist, und weil zweitens der Benutzer des TR440 schon damit vertraut ist.

   Es wurde jedoch auf ein gesondertes Fluchtsymbol verzichtet - der Zeilenvorschub hat diese Funktion - und die Keywordsteuerung wurde nicht implementiert; Spezifikationswerte werden also nur über ihre Position den Spezifikationen zugeordnet.

   Im übrigen wurde versucht, die benutzerfreundlichen features der TR440-Kommandosprache zu übernehmen, z. B. den einheitlichen Aufbau der verschiedenen Befehle, die Formatfreiheit, die bzgl. Sonderzeichen saubere Trennung von Fremdstrings zu den übrigen Spezifikationswerten, und nicht zuletzt die Dialogfähigkeit beim Korrigieren syntaktisch fehlerhafter Spezifikationswerte usw.

Zum Bedienungskomfort des Operators zählt außer den bisher aufgezählten Möglichkeiten insbesondere auch die Gestaltung des eigentlichen dialoggesteuerten Korrigierens.

Das wichtigste ist dabei, möglichst viel Information pro Dialogzyklus eingeben zu können, das heißt in einer einzigen Eingabe in möglichst einfacher und kurzer Form alle gewünschten Korrekturen durchführen zu können, um Reaktionszyklen zu sparen.

Zusätzlich sollten alle Möglichkeiten, die sich durch die Hardware-Eigenschaften der verschiedenen Terminals bieten, voll ausschöpfbar sein.

Aus all diesen Überlegungen heraus entstand das Programm AUFBEREITE , das es nach den bisherigen Erfahrungen gestattet, den für Korrekturen notwendigen Zeitaufwand im Dialog um den Faktor 2 bis 3 gegenüber herkömmlichen Editoren zu reduzieren. Durch seine Möglichkeiten der dialogorientierten Selbstbeschreibung sowie durch seine einfache Syntax ist AUFBEREITE darüber hinaus trotz seiner sehr umfangreichen Leistungen leicht erlernbar.

II EINFÜHRUNG IN DIE BENUTZUNG

Um den Einstieg in die Benutzung der umfangreichen Leistungen von AUFBEREITE zu erleichtern, ist die Einführung mehrstufig gegliedert.

Es wird empfohlen, die jeweils nächste Stufe erst zu studieren, wenn man sich in der vorigen durch Üben "eingelebt" hat.

Die Stufe 1 bietet etwa die Leistungen, die in jedem halbwegs brauchbaren Editor herkömmlicher "Bauart" wiederzufinden sind, der ein satzorientiertes Arbeiten erlaubt.

Stufe 2 bietet Erweiterungen, die schon keineswegs mehr selbstverständlich für Editoren sind, obwohl sie eigentlich unbedingt dazugehören. Nach Kenntnis von Stufe 2 sollte man in der Lage sein, IV verstehen zu können.

Stufe 3 schließlich erlaubt Möglichkeiten, die zum größten Teil in keinem anderen dem Verfasser bekannten Editor enthalten sind.

In dem Kapitel VI wird dann eine zusammenfassende Beschreibung aller Befehle gegeben, die die Kenntnis von V voraussetzt.

STUFE 1

1.) Start von AUFBEREITE

Gestartet wird das Programm entweder mit dem Kommando

□AUFBEREITE,DATEI=<Dateiname>

oder durch

□STARTE,AUFBEREITE,DATEI=1-<Dateiname>

Mit der angegebenen Datei wird dann direkt gearbeitet.

2.) Beendigung von AUFBEREITE

Beenden kann man den Lauf von AUFBEREITE entweder durch den Befehl

S T O P

oder durch eine leere Eingabe.

Befindet man sich nicht in der Grundstufe des Operators, so kann man den Operatorlauf jedenfalls durch mehrmaliges Wiederholen der leeren Eingabe verlassen, die immer unkritisch ist und nichts zerstören kann.

3.) Anhalten von AUFBEREITE

Wenn man etwas Falsches oder Ungeschicktes eingegeben hat, so kann man durch ein einfaches Mittel den Operator dazu veranlassen, auf jeden Fall sofort "alles stehen und liegen zu lassen" und in die Grundstufe zurückkehren, nämlich durch:

□XAN□.

□□ABW□: HALT,AUFBEREITE□.

Auch durch dieses Mittel kann nichts zerstört werden - lediglich der Rest der vorigen Eingabe, der noch nicht abgearbeitet ist, geht verloren.

4.) Verhalten bei unklaren Situationen

Wenn der Operator eine Frage stellt, die man sich nicht erklären kann, so gibt es zwei nützliche Reaktionsmöglichkeiten:

a) Eingabe eines *

Der "*" (also das Multiplikationszeichen) veranlaßt den Operator dazu, eine nähere Erklärung auszudrucken und anschließend an derselben Stelle forzufahren, das heißt, seine letzte Eingabeforderung zu wiederholen.

b) leere Eingabe

Dies empfiehlt sich, wenn man aus der Erklärung immer noch nicht schlau geworden ist und bewirkt ein Übergehen des Befehles, der den Fehler verursachte.

5.) Der Befehl bei AUFBEREITE

Jeder AUFBEREITE-Befehl steht in einer Eingabezeile und hat ähnlich wie bei der TR440-Kommandosprache die Form:

<Befehlsname> <Spezifikationswert>...

Befehlsnamen sind maximal 6 Buchstaben lang und werden entweder ohne Spezifikationswert oder mit Spezifikationswerten, die durch Kommata getrennt werden, eingegeben.

Zwei Kommata hintereinander kennzeichnen einen fehlenden Spezifikationswert, der zu"undefiniert" ergänzt wird, also dieselbe Funktion hat wie ein Minuszeichen. Es gibt eine Reihe von Spezifikationswerten, die aus mehreren Teilwerten bestehen dürfen. Diese werden wie bei der TR440-Kommandosprache durch Apostrophe getrennt.

Leerzeichen sind irrelevant, außer sie treten in Fremdstrings auf, die durch einen Schrägstrich eingeleitet werden. Zeilenwechsel dürfen nur zwischen zwei Befehlen oder in Fremdstrings auftreten.

In einer Eingabe dürfen mehrere Befehle direkt hintereinander eingegeben werden, durch Zeilenwechsel getrennt.

Die im folgenden in Stufe 1 beschriebenen Befehle haben alle als erste Spezifikation BEREICH, die angibt, in welchem Dateibereich oder mit welchen Sätzen etwas "getan" werden soll.

Dabei sind sowohl Einzelsätze als auch Teilbereiche der Datei als Teilwerte (durch Apostrophe getrennt) angebbar.

6.) Protokollieren von Sätzen erreicht man durch den Befehl P.

Beispiele:

P, 1-	Protokolliere alle Sätze der Datei
P, 100-1000	Protokolliere die Sätze 100 bis 1000
P, 100000	Protokolliere den Satz mit der Nummer 100000
P, 1000+10	Protokolliere von Satz 1000 an 10 Sätze
P, 5000+1	Protokolliere den ersten gefundenen Satz, dessen Nummer größer oder gleich 5000 ist
P, 100'1000	Protokolliere die Sätze 100 und 1000
P, 1000+5'3000-	Protokolliere ab Satz 1000 die ersten 5 Sätze und von Satz 3000 ab bis zum Dateiende

7.) Löschen kann man Sätze mit Hilfe des Befehls L.

Beispiele:

L, 2000	Lösche den Satz mit der Nummer 2000
L, 3000-4000	Lösche alle Sätze in dem Bereich von 3000 bis 4000
L, 700+5	Lösche ab Satz 700 die ersten 5 gefundenen Sätze
L, 100'200'300+3	Lösche die Sätze 100, 200 und 3 Sätze ab Satz 300

Sollen die gelöschten Sätze nicht protokolliert werden, so benutzt man dafür den Befehl NL.

Beispiele:

NL, 100-300	Lösche die Sätze 100 bis 300, ohne sie zu protokollieren
NL, -100	Lösche alle Sätze (ohne Protokollierung), deren Nummer kleiner oder gleich 100 ist.

8.) Eintragen von Sätzen mit führender Satznummer erlaubt der Befehl EIN.

Die Sätze selbst werden dabei als Fremdstring angegeben, der erst durch das Eingabeende abgeschlossen wird.

Alle einzelnen Zeilen, die zwischen dem Schrägstrich und dem Eingabeende stehen, stellen dabei jeweils genau einen einzutragenden Satz dar. Jede Zeile beginnt mit einer Satznummer, bei der führende Nullen weggelassen werden können. Danach folgt ein Leerzeichen, das das Ende der Satznummer und den Beginn des Satzinhaltes anzeigt.

Dahinter steht dann der eigentliche Satzinhalt.

Beispiel: siehe nächste Seite

Beispiel: EIN,/

10000␣ erster Satz

25␣zweiter Satz

300␣dritter Satz

810003␣letzter Satz□.

In diesem Beispiel werden also die Sätze 25, 300, 10000und 810003 eingetragen. Die Reihenfolge ist beliebig.

9.) Speichern von Satzfolgen mit angegebener Numerierung ist mit Hilfe des Befehls S möglich.

Die einzeln einzutragenden Sätze werden als jeweils eine Zeile im Fremdstring angegeben. Sie werden ab einer bestimmten Anfangssatznummer mit wählbarer Schrittweite eingetragen. Der Fremdstring wird wiederum durch das Eingabeende abgeschlossen.

Beispiele: S,10000+10,/

Z1

Z2

Z3

Z4□.

Hierdurch werden die Sätze 10000, 10010, 10020, 10030 und 10040 eingetragen und protokolliert.

S,✱H+100,/

X

Y

Z□.

Hierdurch werden mit der Schrittweite 100 drei Sätze eingetragen, und zwar hinter den letzten Satz der Datei.
Hatte der letzte Satz der Datei z. B. die Nummer 90, so werden die Sätze 100,200 und 300 eingetragen.
Wird als Anfangssatznummer die Zeichenfolge ✱H genommen, so beginnt die Numerierung bei der bisher höchsten existierenden Satznummer plus der aktuellen Schrittweite.

10.) Umnumerieren von Sätzen der Datei kann man mit dem Befehl NUM.

Er sollte jedoch nicht dazu benutzt werden, große Teile einer Datei oder gar die ganze Datei umzunumerieren - dazu benutze man die vorhandenen Programmiersystemkommandos (TNUMERIERE), da dieser Befehl im Kernspeicher zwischenpuffert.

Beispiele:

NUM,100+10,10-40	Numeriere die Sätze 10 bis 40 um mit der Anfangsnummer 100 und der Schrittweite 10.
NUM,1000+10,1000+30	Numeriere die ersten 30 Sätze ab Satz 1000 mit der Schrittweite 10 um, beginnend bei Satz 1000.
NUM,1010,1040	Numeriere den Satz 1040 in 1010 um.
NUM,10000+20,2000-	Numeriere alle Sätze ab Satz 2000 um mit der Anfangssatznummer 10000 und der Schrittweite 20.

11.) Vertauschen von Sätzen geschieht ebenfalls mit dem Befehl NUM, da Quell- und Ziel-Bereiche sich beliebig überschneiden dürfen (im Gegensatz zu TNUMERIERE).

Beispiele:

NUM,1000+100,1100'1000	Vertausche die Sätze 1000 und 1100.
NUM,33+1,36-40'33-35	Vertausche die Bereiche (33 bis 35) und (36 bis 40).

12.) Duplizieren von Sätzen geschieht mit dem Befehl D .

Auch hierbei ist zu beachten, daß alle Sätze im Kernspeicher zwischengepuffert werden.

Beispiele:

D,1000,100	Dupliziere den Satz mit der Nummer 100 in den Satz 1000.
D,100+2,1-50	Dupliziere die Sätze 1 bis 50, beginnend bei Satz 100, mit der Schrittweite 2.
D,1000+20,17'5'10	Dupliziere die Sätze 17, 5 und 10 in der angegebenen Reihenfolge ab Satz 1000 mit der Schrittweite 20.

13.) Korrigieren kann man Sätze auf zwei grundsätzliche verschiedene Arten, je nachdem, ob das Terminal über einen Blockmodus verfügt oder nicht, mit dem Befehl K.

Hier soll nur das fernschreiberanaloge Korrigieren ohne Blockmodus, wie es herkömmliche Editoren kennen, beschrieben werden.

Arbeitet man an einem SIG50, SIG51, SIG100 oder an einem VISTAR, so gebe man einmal zu Beginn des AUFBEREITE-Laufes den Befehl "MODUS, FSR".

Beim Korrigieren werden alle Sätze nacheinander zur Korrektur vorgelegt, die im K-Befehl angegeben wurden.

Beispiel: K,100'1000'23400+5'150000-

Hierbei werden nacheinander die Sätze 100, 1000, ab Satz 23400 5 Sätze und von Satz 150000 bis zum Dateiende zur Korrektur vorgelegt, und zwar in der Form:

```
100␣DIES IST SATZ HUNDERT
␣␣:
```

Das erste eingegebene Zeichen kommt also unter das nicht zum Satz gehörende Blank hinter der Satznummer zu stehen.

Dieses erste eingegebene Zeichen hat nämlich eine Sonderbedeutung:
Es definiert das Identitätszeichen für den jeweiligen Korrekturvorgang.
Zur Erklärung des Korrekturmechanismus:

> Jedes eingegebene Zeichen bezieht sich auf das jeweils darüberstehende Zeichen des vorgelegten Satzes.
> Es gibt 3 Sonderzeichen - alle anderen Eingabezeichen ersetzen das darüberstehende Zeichen.
>
> Diese Sonderzeichen sind:
>
> das Identitätszeichen,
> das Löschzeichen und
> das Ersetzungszeichen.

Das Identitätszeichen wird wie gesagt bei jeder Korrektureingabe neu definiert durch das erste eingegebene Zeichen und darf nicht identisch mit einem der beiden anderen Sonderzeichen sein.

Alle Zeichen des vorgelegten Satzes, unter denen das Identitätszeichen eingegeben wird, bleiben erhalten.

Alle Zeichen, unter denen das Löschzeichen eingegeben wird, werden gelöscht. Wenn das Löschzeichen außerdem das letzte Zeichen der ersten Eingabezeile ist, wird der gesamte Rest des vorgelegten Satzes gelöscht.

Das Löschzeichen ist folgendermaßen voreingestellt:

Gerät	Löschzeichen
SIG100, VISTAR	> ("Größer-Zeichen")
8-kanal-FSR	} ("geschweifte Klammer zu")
5-kanal-FSR	] ("eckige Klammer zu")

Es kann jedoch auch dynamisch umgestellt werden (Befehl LOEZEI).

Wir nehmen in den folgenden Besipielen an, es handele sich um den einfach Fall eines 5-kanal-Fernschreibers mit dem Löschzeichen "]".

Beispiele:

Ein- und Ausgaben	
K, 100␍.	korrigiere Satz 100
100 DIES IST SATZ HUNDERT	Satz vorlegen
␣␍:.....WAR␍.	"." ist das Identitätszeichen
100 DIES WAR SATZ HUNDERT	Satz geändert neu vorlegen
␣␍:.....]]]].␍.	löschen von "Satz "
100 DIES WAR HUNDERT	erneute Vorlage
␣␍:.....MIST]␍.	Löschzeichen ist letztes Zeichen
100 DIES WAR MIST	Rest wird abgeschnitten
␣␍:␍.	leere Eingabe, Korrektur fertig
100 DIES WAR MIST	Protokollierung des fertigen Satzes
BEFEHL␍:	Korrektur beendet - Anfrage nach neuen Befehlen

In diesem Beispiel wurde immer der Punkt als Identitätszeichen gewählt.

Das dritte Sonderzeichen, das Ersetzungszeichen, ist auch dynamisch umstellbar (Befehl ERSZEI)- seine Voreinstellung ist:

Gerät	Ersetzungszeichen
SIG100,VISTAR	< ("kleiner-Zeichen")
8-kanal-FSR	{ ("geschweifte Klammer auf")
5-kanal-FSR	[(eckige Klammer auf")

Alle Zeichen, unter denen das Ersetzungszeichen steht, werden ersetzt, und zwar durch die nächste komplette Zeile der Eingabe.
Sind mehrere Ersetzungszeichen in der ersten Eingabezeile, so werden die darüberstehenden Zeichen jeweils durch die nächste noch nicht abgearbeitete Zeile der Eingabe ersetzt.

Wenn mehr Ersetzungszeichen als Zeilen derselben Eingabe vorhanden sind, so wird die letzte Eingabezeile für die restlichen Ersetzungszeichen genommen.

Beispiel:

EIN- UND AUSGABEN	BEDEUTUNG
K,100'200↵.	Korrigiere die Sätze 100 und 200
100 DIES WAR MIST	Satz vorlegen
␣↵:.....[	
␣KEIN␣↵.	1 Ersetzung
100 DIES WAR KEIN MIST	erneute Vorlage
␣↵: ..AS[...[....[	3 Ersetzungen
␣HIER␣	1. Ersetzung
␣AUCH␣	2. Ersetzung
␣BESONDERER␣↵.	3. Ersetzung
100 DAS HIER WAR AUCH KEIN BESONDERER MIST	erneute Vorlage
␣↵:↵.	leere Eingabe
100 DAS HIER WAR AUCH KEIN BESONDERER MIST	
200 DET WIERD ABBER WELCHEN GELL !	Vorlage des nächsten Satzes
␣↵:..AS ..]....].....R.]]][	5 Löschungen, 1 Ersetzung
NICHT WAHR	das ist die Ersetzung
↵.	Leerzeile = Korrektur des Satzes beenden
200 DAS WIRD ABER WELCHER NICHT WAHR !	Protokollieren des endgültigen Satzes
BEFEHL↵:	

Eine einzelne Leerzeile in der Eingabe, z. B. durch eine leere Eingabe, bewirkt die Beendigung der Korrektur des gerade bearbeiteten Satzes und die Vorlage des nächsten Satzes, wenn von dem K-Befehl her noch einer zur Korrektur ansteht - sonst werden wieder Befehle erfragt. Werden zwei oder mehr Leerzeilen hintereinander eingegeben, so wird die Abarbeitung des gesamten K-Befehls beendet - noch anstehende Korrektursätze werden ignoriert.

In die Datei zurückgetragen wird ein Satz erst, wenn seine Korrektur durch mindestens eine Leerzeile beendet wird. Es ist ersichtlich, daß in einer einzigen Eingabe beliebig viele Korrekturen durch Löschen von Zeichen, durch Darunterschreiben von irgendwelchen anderen Zeichen oder durch Ersetzungen (wenn die neue Zeichenfolge länger als die alte ist) durchführbar sind.

Gleichzeitig ist aus Gründen der Übersichtlichkeit festgelegt, daß eine komplette Eingabe nur einen einzigen Satz verändert, daß also nicht auswertbare Zeilen ignoriert werden (wenn z. B. mehr Zeilen folgen als Ersetzungszeichen in der ersten Eingabezeile standen).

Die 3 Sonderzeichen haben darüber hinaus nur in der ersten Eingabezeile eine spezielle Funktion - in allen weiteren Eingabezeilen werden sie wie normale Textzeichen behandelt.

Weitere Beispiele:

```
K,1000-¤.
10010 DER SATZ IST NICHTS
␣␣␣¤:.J¤.                    (Satz wird gelöscht, da nach Korrektur leer)
L 10010 DER SATZ IST NICHTS
10500 ANTON UND BERTA
␣␣␣¤:..[..[..[
␣␣␣                          (3-mal dieselbe Ersetzung)
¤.
10500 A␣␣␣TO␣␣␣U␣␣␣D BERTA
12000 X=3.14158285 ;
␣␣␣¤;!!C!!!!!!9!C            ("!" ist Identitätszeichen)
:=
653
                             (Leerzeile)
¤.                           (Abbruch Korrektur-Befehl)
12000 X:=3.1415926535 ;
BEFEHL¤:
```

14.) Sitzungsbeispiel

Das folgende Sitzungsbeispiel enthält bereits Befehle und andere Möglichkeiten, die bisher noch nicht beschrieben wurden.

Es dient dazu, das bisherige besser zu verstehen und zu Stufe 2 und 3 einen leichteren Einstieg zu vermitteln. Man lese es sich deshalb sorgfältig durch.

```
#EINSCHLEUSE,N.DEF,LFD,,LESEN
#NAME=N.COBOLKOPF
#TDEKL.,COBOLDATEI

#AUFBEREITE,COBOLDATEI,MODUS=FSR,DEFIN.=30,PROTOKOLL=£'A'KD#.
EINGESCHL.: DEF(0001.00)  KAT: N
EINGESCHL.: COBOLKOPF(0001.00)  KAT: N
KREIERT: COBOLDATEI(0001.00)


*** Datei = &STDDB.COBOLDATEI(0001.00), RAM, U800, *leer*


Befehl#:LOEZEI,>#.
Befehl#:ERSZEI,<#.
Befehl#:ERRZEI,168#.
Befehl#:GET,COBOLKOPF,100+100,1-#.
  100         IDENTIFICATION DIVISION.
  200         PROGRAM-ID. *******.
  300         ENVIRONMENT DIVISION.
  400         CONFIGURATION SECTION.
  500         SOURCE-COMPUTER. TR 440.
  600         OBJECT-COMPUTER. TR 440.
  700         SPECIAL-NAMES.
  800             PAGE IS NPAGE.
  900         INPUT-OUTPUT SECTION.
 1000          FILE-CONTROL.
 1100              SELECT ********** ASSIGN TO ***_**
 1200              ACCESS MODE IS SEQUENTIAL
 1300              PROCESSING MODE IS SEQUENTIAL.
 1400        /
 1500         DATA DIVISION.
 1600         FILE SECTION.
Befehl#:K,1-,12-,/***#.
0200        PROGRAM-ID. *******.
  #:                    BEISPIEL.
#.
0200        PROGRAM-ID. BEISPIEL.
1100            SELECT ********** ASSIGN TO ***_**
  #:                      EINGBAE>>>          WSP-01#.
1100            SELECT EINGBAE ASSIGN TO WSP-01
  #:                         AB#.
#.
1100            SELECT EINGABE ASSIGN TO WSP-01
Befehl#:NUM,10000+100,1400-
GET,COBOLKOPF,1400+100,2000-2200
K,1400-,12-,/***#.
N 10000        /
N 10100         DATA DIVISION.
N 10200         FILE SECTION.
 01400              SELECT ********** ASSIGN TO ***_**
 01500              ACCESS MODE IS SEQUENTIAL
 01600              PROCESSING MODE IS SEQUENTIAL.
01400               SELECT ********** ASSIGN TO ***_**
  #:                       AUSGABE>>>#.
01400               SELECT AUSGABE
  #: )#.
01400               SELECT AUSGABE
  #:>#.
01400               SELECT ********** ASSIGN TO ***_**
  #:                       AUSGABE>>>          WSP-03
#.
01400               SELECT AUSGABE ASSIGN TO WSP-03
Befehl#:ND,1700+100,1100-1600#.
Befehl#:K,1400-,12-,/SELECT#.
01400               SELECT AUSGABE ASSIGN TO WSP-03
  #:                       <>>>>>          WSP-04
RAMDATEI#.
01400               SELECT RAMDATEI ASSIGN TO WSP-04
  #:#.
01400               SELECT RAMDATEI ASSIGN TO WSP-04
01700               SELECT EINGABE ASSIGN TO WSP-01
  #:                       DRUCKD<          SDR 05
ATEI#.
01700               SELECT DRUCKDATEI ASSIGN TO SDR-05
  #:
BREAK
P,1-#.
```

```
01700           SELECT DRUCKDATEI ASSIGN TO SDR-05
00100         IDENTIFICATION DIVISION.
00200         PROGRAM-ID. BEISPIEL.
00300         ENVIRONMENT DIVISION.
00400         CONFIGURATION SECTION.
00500         SOURCE-COMPUTER. TR 440.
00600         OBJECT-COMPUTER. TR 440.
00700         SPECIAL-NAMES.
00800             PAGE IS NPAGE.
00900         INPUT-OUTPUT SECTION.
01000         FILE-CONTROL.
01100             SELECT EINGABE ASSIGN TO WSP-01
01200             ACCESS MODE IS SEQUENTIAL
01300             PROCESSING MODE IS SEQUENTIAL.
01400             SELECT RAMDATEI ASSIGN TO WSP-04
01500             ACCESS MODE IS SEQUENTIAL
01600             PROCESSING MODE IS SEQUENTIAL.
01700             SELECT DRUCKDATEI ASSIGN TO SDR-05
01800             ACCESS MODE IS SEQUENTIAL
01900             PROCESSING MODE IS SEQUENTIAL.
02000             SELECT AUSGABE ASSIGN TO WSP-03
02100             ACCESS MODE IS SEQUENTIAL
02200             PROCESSING MODE IS SEQUENTIAL.
10000       /
10100         DATA DIVISION.
10200         FILE SECTION.
Befehl#:K,1700#.
01700           SELECT DRUCKDATEI ASSIGN TO SDR-05
  #:<05<02#.
01700           SELECT DRUCKDATEI ASSIGN TO SDR-02
  #:
BREAK
TUE,DEF,1-1000
INF,SIT#.
01700           SELECT DRUCKDATEI ASSIGN TO SDR-02


MODUS,FSR
SYSI,-
AEND,-
ZEILBR,161
KPROT,-
KORBER,-
KRIT,-STD-
PROT,E'A'KO
ZENDE,#021
MAL,53
LOEZEI,#078
ERSZEI,#101
ERRZEI,#044
MAXDEF,30
SATZLG,161
SNRLNG,5
TRIM,-
WANDEL,-STD-
BER, - 999999


*** TABULATOREN : ***

:  36 45 50


*** Datei = &STDDB.COBOLDATEI(0001.00), RAM, U800, 25 S, 100 - 10200


***  aktive Ersetzungen :  ***

*1  =  1,/.A.
*2  =  1,/.B.
*3  =  1,/.C.
*4  =  1,/.D.
*5  =  1,/.E.
*6  =  1,/.F.
*7  =  1-161,/.,P.;PICTURE IS
*8  =  1-161,/.,F. FILLER;PICTURE IS X(
*9  =  1-161,/.,S.;VALUE SPACES
*10  =  1-161,/.,O. SELECT
*11  =  1-161,/.,LS. LABEL RECORDS ARE STANDARD
*12  =  1-161,/.,LO. LABEL RECORDS ARE OMITTED
```

```
#13  =  1-161,/.,.
#14  =  1-161,/.,A. ASSIGN TO
#15  =  1-161,/.,R. DATA RECORD IS
#16  =  1,/.1.       01
#17  =  1,/.2.           02
#18  =  1,/.3.               03
#19  =  1,/.7.       77
#20  =  1-161,/.,V.;VALUE IS
#21  =  1-161,/.,O.;OCCURS
#22  =  1-161,/.,M.;RECORDING MODE IS
#23  =  1-161,/.[.;PICTURE IS X(
#24  =  1-161,/.].).
#25  =  1-161,/.<.;PICTURE IS 9(
#26  =  1,/.*.      *    ************************************************
#27  =  1-161,/.,T.;REDEFINES
#28  =  1,/.8.             88
#29  =  1,/.4.                 04



Befehl#:S,10300+100,/
AFD EINGABE,LS.
1 ESATZ.
2 TEIL1[8]
2,F10).
2 NAMEN[50]
2 ORT.
3 PLZ<4).#.
  10300        FD EINGABE LABEL RECORDS ARE STANDARD.
  10400        01    ESATZ.
  10500            02  TEIL1              PICTURE IS X(8).
  10600            02  FILLER             PICTURE IS X(10).
  10700            02  NAMEN              PICTURE IS X(50).
  10800            02  ORT.
  10900                03  PLZ            PICTURE IS 9(4).
Befehl#:S,,/
3 STRASSE[20]
3 BERSCHL<6).
AFD AUSGABE,LS.
1 ASATZ[100]
AFD RAMDATYI,LS.
1 HSATZ.
2 AKEY<13) COMP.
2 BERUF[80]
AFDDRUCKDATEILSD#.
  11000                03  STRASSE       PICTURE IS X(20).
  11100                03  BERSCHL       PICTURE IS 9(6).
  11200        FD AUSGABE LABEL RECORDS ARE STANDARD.
  11300        01    ASATZ                PICTURE IS X(100).
  11400        FD RAMDATYI LABEL RECORDS ARE STANDARD.
  11500        01    HSATZ.
  11600            02  AKEY               PICTURE IS 9(13) COMP.
  11700            02  BERUF              PICTURE IS X(80).
  11800        FDDRUCKDATEILD
Befehl#:K,11400#.
11400        FD RAMDATYI LABEL RECORDS ARE STANDARD.
  #:<Y<E
#.
11400        FD RAMDATEI LABEL RECORDS ARE STANDARD.
Befehl#:K,11800#.
11800        FDDRUCKDATEILD
  #:           DRUCKDATEI,LD.#.
11800        FDDDRUCKDATEI,LD.
  #:++++++++++ +
#.
11800        FD DRUCKDATEI,LD.
Befehl#:E,11800#.
E 11800      FD DRUCKDATEI LABEL RECORDS ARE OMITTED.
*** Dateiende
Befehl#:P,11800-#.
11800        FD DRUCKDATEI LABEL RECORDS ARE OMITTED.
Befehl#:S,,/
1DRUSATZ[161]
AWORKING-STORAGE SECTION.
7H1[10]
1KOFELD.
2K02.
3K021[30]
```

```
3K022<6).
2K03[40]
2K04[20]#.
  11900         01   DRUSATZ                    PICTURE IS X(161).
  12000         WORKING-STORAGE SECTION.
  12100         77   H1                         PICTURE IS X(10).
  12200         01   KOFELD.
  12300             02 K02.
  12400                03 K021                  PICTURE IS X(30).
  12500                03 K022                  PICTURE IS 9(6).
  12600             02 K03                      PICTURE IS X(40).
  12700             02 K04                      PICTURE IS X(20).
Befehl#:TUE,DEF,1000-1999
INF,SIT#.


MODUS,FSR
SYSI,-
AEND,-
ZEILBR,161
KPROT,-
KORBER,-
KRIT,-STD-
PROT,E'A'KO
ZENDE,#021
MAL,53
LOEZEI,>
ERSZEI,<
ERRZEI,#044
MAXDEF,30
SATZLG,161
SNRLNG,5
TRIM,-
WANDEL,-STD-
BER, - 999999


*** TABULATOREN : ***

:   12 14 18 25 30 35 40 50


*** Datei = &STDDB.CDBOLDATEI(0001.00), RAM, U800, 50 5, 100 - 12700


***  aktive Ersetzungen :  ***

*1  =  1,/.A.
*2  =  1,/.B.
*3  =  1,/.C.
*4  =  1,/.D.
*5  =  1,/.E.
*6  =  1,/.F.
*7  =  1-161,/.,D. DISPLAY
*8  =  1-161,/.,UC. UPON CONSOLE
*9  =  1-161,/.,A. ALTER
*10  =  1-161,/.,T. TO PROCEED TO
*11  =  1-161,/.,P. PERFORM
*12  =  1-161,/.,M. MOVE
*13  =  1-161,/.,W. WRITE
*14  =  1-161,/.,R. READ
*15  =  1-161,/.,F. AFTER
*16  =  1-161,/.,IK. INVALID KEY
*17  =  1-161,/.,G. GO TO
*18  =  1-161,/.,OI. OPEN INPUT
*19  =  1-161,/.,OO. OPEN OUTPUT
*20  =  1,/.*.      *    ************************************************
*21  =  1-161,/.[. TO
*22  =  1-161,/.>. GREATER THAN
*23  =  1-161,/.<. LESS THAN
*24  =  1-161,/.,C. ACCEPT
*25  =  1-161,/.7.IF
*26  =  1-161,/.,E. ENTER
*27  =  1-161,/.(. (
*28  =  1-161,/.=. =
*29  =  1-161,/.,+. ADD
*30  =  1-161,/.,-. SUBTRACT
```

```
Befehl#:S,,/
APROCEDURE DIVISION.
AHPSECTION.
AOEFFNEN.
B,OIEINGABE
B,OOAUSGABE DRUCKDATEI
BOPEN I-O RAMDATEI.
ALOOP.#.
  12800         PROCEDURE DIVISION.
  12900         HPSECTION.
  13000         OEFFNEN.
  13100              OPEN INPUT EINGABE
  13200              OPEN OUTPUT AUSGABE DRUCKDATEI
  13300             OPEN I-O RAMDATEI.
  13400         LOOP.
Befehl#:K,12900#.
12900         HPSECTION.
   #:<P<P
#.
12900         HP SECTION.
Befehl#:S,,/
B,REINGABE RECORD A T END
CCLOSE EINGABE RAMDATEI DRUCKDATEI AUSGABE STOP RUN.
B,PWANDELN
,GLOOP.#.
  13500              READ EINGABE RECORD A T END
  13600                 CLOSE EINGABE RAMDATEI DRUCKDATEI AUSGABE STOP RUN.
  13700              PERFORM WANDELN
  13800  GO TO LOOP.
Befehl#:K,13500'13800#.
13500              READ EINGABE RECORD A T END
   #:                                  <#.
13500              READ EINGABE RECORD A
   #:<A<AT END
#.
13500              REAT ENDD EINGAT ENDBE RECORD AT END
13800  GO TO LOOP.
   #:<G<            G
BREAK
K,13500#.
13800              GO TO LOOP.
13500              REAT ENDD EINGAT ENDBE RECORD AT END
   #:<AT END<A#.
13500              READ EINGABE RECORD A
   #:                                  T END
#.
13500              READ EINGABE RECORD AT END
Befehl#:S,,/
AWANDELN SECTION.
AANF-WANDELN.
B,MBERSCHL TO AKEY
B,RRAMDATEI RECORD,IK
BC,PIVKRAM.
B,DNAMEN ' BERUF= -' BERUF.
B,GEND-WANDELN.
AIVKRAM.
B,D'KEIN SATZ MIT NR. ' AKEY.
AEND-WANDELN.
B,D#.
  13900         WANDELN SECTION.
  14000         ANF-WANDELN.
  14100              MOVE BERSCHL TO AKEY
  14200              READ RAMDATEI RECORD INVALID KEY
  14300                  PERFORM IVKRAM.
  14400              DISPLAY NAMEN ' BERUF =  ' BERUF.
  14500              GO TO END-WANDELN.
  14600         IVKRAM.
  14700              DISPLAY 'KEIN SATZ MIT NR. ' AKEY.
  14800         END-WANDELN.
  14900              DISPLAY
Befehl#:K,14900#.
14900              DISPLAY
   #:              EXIT.>
#.
14900              EXIT.
Befehl#:EIN,/
1 #UEBERSETZE,SPRACHE=COBOL,QUELLE=/#.

++++LEERE QUELLE ODER FLUCHTSYMBOL AM QUELLANFANG
ENDE PS$COBOLCOM2 (15.01)  0.30
```

```
+++++OPERATORLAUF MIT FEHLER BEENDET: PS&COBOLCOM2

GIB WEITERES VORRANGKOMMANDO#:#.
VORRANGKOMMANDOS AUSGEFUEHRT
Befehl#:EIN#.
Text#:
1 #UEBERSETZE,SPRACHE=COBOL,QUELLE=/
BREAK
S,,/
#SPRINGE,FEHLER,(FE1)
#MONTIERE,BEISPIEL
#STARTE,BEISPIEL,DATEI=1-A'2-B'3-C'4-D#.
  00001 #UEBERSETZE,SPRACHE = COBOL,QUELLE = /
  15000 #SPRINGE AFTER EHLER, (FE1)
  15100 #MONTIERE,BEISPIEL
  15200 #STARTE,BEISPIEL DISPLAY ATEI = 1-A'2-B'3-C'4-D
Befehl#:K,15000-#.
15000 #SPRINGE AFTER EHLER,  (FE1)
   #:          ,> >>>>#.
15000 #SPRINGE,F
   #:>#.
15000 #SPRINGE AFTER EHLER,  (FE1)
   #:          ,> >>>>E
#.
15000 #SPRINGE,FEHLER,  (FE1)
15100 #MONTIERE,BEISPIEL
   #:#.
15200 #STARTE,BEISPIEL DISPLAY ATEI  =  1-A'2-B'3-C'4-D
   #:                , >>>>>>>   >>=#.
15200 #STARTE,BEISPIEL,DATEI=  1-A'2-B'3-C'4-D
   #:#.
15200 #STARTE,BEISPIEL,DATEI=  1-A'2-B'3-C'4-D
Befehl#:ERS
DEF,1,73-80,/&&BEISPIEL
ERS,1
NE,1-#.
*1  =  73-90,/&&BEISPIEL
*** Dateiende
Befehl#:B,BEISP.(RP)#.
```

Beispiele :

```
RP,*H
     Protokolliere den letzten Satz der Datei

RP,*H+5,,/MAX
     Protokolliere die letzten 5 Saetze der
     Datei, in denen die Zeichenfolge ''MAX''
     vorkommt.

RP,100000+3,73-
     Protokolliere die letzten 3 Saetze von
     Satz 100000 an ruckwaerts, die laenger
     als 72 Spalten sind.


Befehl#:RP,*H+2#.
15200 #STARTE,BEISPIEL,DATEI=  1-A'2-B'3-C'4-D                    BEISPIEL
15100 #MONTIERE,BEISPIEL                                          BEISPIEL
Befehl#:K,1-,,/#053#.
00001 #UEBERSETZE,SPRACHE = COBOL,QUELLE = /                      BEISPIEL
   #:++++++++++++++++++++++++++++++++++++>
#.
00001 #UEBERSETZE,SPRACHE = COBOL,QUELLE = /
15000 #SPRINGE,FEHLER,  (FE1)                                     BEISPIEL
   #:++++++++++++++++++++++++++++>
#.
15000 #SPRINGE,FEHLER,  (FE1)
15100 #MONTIERE,BEISPIEL                                          BEISPIEL
   #:++++++++++++++++++>
#.
15100 #MONTIERE,BEISPIEL
15200 #STARTE,BEISPIEL,DATEI=  1-A'2-B'3-C'4-D                    BEISPIEL
   #:                                        >
BREAK
S,,/
#053*FEHLER*#.
```

```
15200 #STARTE,BEISPIEL,DATEI=  1-A'2-B'3-C'4-D
  15300 #*FEHLER*                                                    BEISPIEL
Befehl#:K,15300#.
15300 #*FEHLER*                                                    BEISPIEL
   #:           >
#.
15300 #*FEHLER*
Befehl#:#TU,COBOLDATEI#.

ENDE TU (17.05)  0.39

ANFANG PROTOKOLL
000650              READ RAMDATEI RECORD INVALID KEY
                                          ////////////////////////
                    BEISPIEL
       ////////////////////////
  **FEHLER: READ-FORMAT FUER SEQUENTIAL ACCESS FILE UNZULAESSIG
000660                 PERFORM IVKRAM.
       ///////////////
                    BEISPIEL
ENDE PROTOKOLL

ENDE PS&COBOLCOM3 (15.01)  1.27

++++OPERATORLAUF MIT FEHLER BEENDET: PS&COBOLCOM3

GIB WEITERES VORRANGKOMMANDO#:#ENDE#.
Befehl#:SUCH,1+1,12-,/RAMDATEI
P,*A+3#.
01400               SELECT RAMDATEI ASSIGN TO WSP-04                BEISPIEL
01500               ACCESS MODE IS SEQUENTIAL                       BEISPIEL
01600               PROCESSING MODE IS SEQUENTIAL.                  BEISPIEL
Befehl#:K,1500#.
01500               ACCESS MODE IS SEQUENTIAL                       BEISPIEL
   #:                          RANDOM       #.
01500               ACCESS MODE IS RANDOMTIAL                       BEISPIEL
   #:                                 M>>>>
BREAK
EIN,/
1510 BACTUAL KEY IS AKEY#.
01500               ACCESS MODE IS RANDOM
  01510 BACTUAL KEY IS AKEY                                          BEISPIEL
Befehl#:NERS,-STD-
E,1510#.
E 01510               ACTUAL KEY IS AKEY                                        BEISPIEL
Befehl#:K,1510#.
01510               ACTUAL KEY IS AKEY                              BEISPIEL  BEISPIEL
   #:                              >
BREAK
ERS
SUCH,1511+1,12-,/AKEY
P,*A
SUCH,1500-,7-,/WORKING
P,*A#.
01510               ACTUAL KEY IS AKEY
11600               02  AKEY                PICTURE IS 9(13) COMP.  BEISPIEL
12000          WORKING-STORAGE SECTION.                               BEISPIEL
Befehl#:NUM,12001,11600
K,12001#.
N 12001               02  AKEY                PICTURE IS 9(13) COMP.  BEISPIEL
12001               02  AKEY                PICTURE IS 9(13) COMP.  BEISPIEL
   #:++++++++77++  ++A
#.
12001         77      AKEY                 PICTURE IS 9(13) COMP.  BEISPIEL
Befehl#:#TU,COBOLDATEI#.

ENDE TU (17.05)  0.43

MD BEISPIEL WURDE ERZEUGT
KEINE SYNTAXFEHLER
ERZEUGTES MD  BEISPIEL

ENDE PS&COBOLCOM3 (15.01)  1.77

ENDE MONTIERE (21.02)  2.54

START BEISPIEL

* FEHLER  :  DATEI(STARTE) = 1 - A,
             UNTER DIESEM NAMEN IST IN DER DATENBASIS &STDDB KEINE DATEI BEKANNT.
```

```
* FEHLER  :  DATEI(STARTE) = 4 - D,
            UNTER DIESEM NAMEN IST IN DER DATENBASIS &STDDB KEINE DATEI BEKANNT.

* FEHLER  :  DATEI(STARTE) = 2 - B,
            UNTER DIESEM NAMEN IST IN DER DATENBASIS &STDDB KEINE DATEI BEKANNT.

* FEHLER  :  DATEI(STARTE) = 3 - C,
            UNTER DIESEM NAMEN IST IN DER DATENBASIS &STDDB KEINE DATEI BEKANNT.

***PROGRAMMABBRUCH*** ,WEITERE VERARBEITUNG WEGEN OBEN GENANNTER FEHLER UNMOEGLICH.

ENDE BEISPIEL  0.07

++++OPERATORLAUF MIT FEHLER BEENDET: BEISPIEL

GIB WEITERES VORRANGKOMMANDO#:#ENDE#.
Befehl#:
```

STUFE 2

1.) Die Suchbedingung

Bisher war nur die Rede vom satzorientierten Zugriff auf die Datei, das heißt, bei den Befehlen wurden Satznummern angegeben, und mit den so spezifizierten Sätzen wurde irgendetwas gemacht:

sie wurden protokolliert, gelöscht, korrigiert etc.

Sehr häufig braucht man aber den assoziativen Zugriff, also den Zugriff auf Sätze durch Spezifizierung ihres Inhalts in der Art:

Protokolliere alle Sätze, in denen die Zeichenfolge XY vorkommt oder:
Lösche alle Sätze, in denen ein Punkt vorkommt oder:
Korrigiere alle Sätze, die länger als 72 Zeichen sind,
usw.

Solche assoziativen Zugriffe werden bei allen Befehlen, die überhaupt irgendetwas mit Sätzen "tun", die also z. B. keine globalen Umsteuerbefehle sind, durch die zweite und dritte Spezifikation gesteuert (bei NUM,D und GET die 3. und 4. Spezifikation).

Diese beiden Spezifikationen heißen

SPALTENBEREICH und SUCHSTRING

und werden mit der 1. Spezifikation BEREICH, die wir bereits kennen, unter dem Begriff <Zugriff> zusammengefaßt. Unter SUCHSTRING wird ein Fremdstring angegeben (durch einen Schrägstrich eingeleitet), der durch das Zeilenende abgeschlossen wird.

Die beiden Spezifikationen sind optional.
Folgende 4 Fälle sind möglich:

a) SPALTENBEREICH und SUCHSTRING sind beide undefiniert.
Dann haben wir den bereits bekannten Fall des reinen satzorientierten Zugriffs.

b) SUCHSTRING ist angegeben.
Dann trifft die Suchbedingung auf alle Sätze zu, die die Zeichenfolge in SUCHSTRING enthalten.

Beispiel:

P,1-,,/MAX	Protokolliere alle Sätze in denen die Zeichenfolge "MAX" vorkommt.
L,10-100,,/MIST	Lösche alle Sätze von 10 bis 100, die die Zeichenfolge "MIST" enthalten.

c) SPALTENBEREICH und SUCHSTRING sind angegeben

Dann trifft die Suchbedingung auf alle Sätze zu, die die angegebene Zeichenfolge in dem spezifizierten Spaltenbereich enthalten.

Beispiel:

K,10000+5, 3-7,/?	Korrigiere die ersten 5 Sätze ab Satz 10000, die irgendwo in Spalte 3 bis 7 ein Fragezeichen enthalten.
P,-1000,6,/␣	Protokolliere alle Sätze bis Satz 1000, die in Spalte 6 ein Leerzeichen enthalten.

d) Nur SPALTENBEREICH ist angegeben.

Dann erfüllen alle Sätze die Suchbedingung, deren Länge in dem angegebenen Spaltenbereich liegt.

Beispiel:

P,1-,73-	Protokolliere alle Sätze, die 73 Zeichen lang oder länger sind.
K,1-,5-10	Korrigiere alle Sätze, deren Länge zwischen 5 und 10 Zeichen liegt.

2.) Tabulatoren

Sie wirken nicht nur bei der Eingabe, sondern bei allen dateiverändernden Befehlen.

Zu Beginn des Operatorlaufes von AUFBEREITE existieren noch keine Tabulatoren - sie können dynamisch mit Hilfe des Befehls TAB definiert bzw. auch gelöscht werden.

Die Tabulatorfunktion ist folgende:

Es können mehrere (max. 20) Tabulatorpositionen definiert werden sowie ein Spezialzeichen, bei dessen Auftreten im Satz durch Auffüllen mit Leerzeichen auf die nächste Tabulatorposition weitergeschaltet wird.

Beispiel für das Definieren von Tabulatoren:

```
TAB,10'20'30'40,/&
```

Dadurch werden die Positionen 10,20,30 und 40 als Tabulatorpositionen definiert, und das "&" wird als Tabulatorzeichen festgelegt, bei dessen Auftreten zur nächsten Position innerhalb des Satzes fortgeschaltet werden soll.

Beispiel für die Wirkung dieser Tabulatoren-Definition:

Eingabe:

```
S,100+10,/
SPALTE1&SPALTE10&&SPALTE30
&A&B&C
&&A&B.
```

Protokollierung der Wirkung:

100	SPALTE1	SPALTE10	SPALTE30
110		A B	C
120		A	B

Durch den Befehl TAB ohne Spezifikationen werden alle Tabulatordefinitionen wieder gelöscht. Bei jedem neuen TAB-Befehl werden alle Tabulatordefinitionen und das Tabulatorzeichen neu definiert, alle evtl. vorher gültigen Tabulatoren werden ungültig.

3.) Informieredienste

Es gibt 2 Befehle, mit denen man sich Informationen ausgeben lassen kann.

a) Der Befehl INF

Mit INF, SIT kann man sich alle aktuellen (dynamisch änderbaren) globalen Einstellungen ausdrucken lassen, also z. B.:

Modus, Zeilenbreite, Ersetzungszeichen, Löschzeichen, aktuelle Datei, Tabulatoren, Ersetzungen etc.

INF,INF liefert alle als Modus bei INF zulässigen Spezifikationswerte

INF,BEF liefert eine Liste aller zulässigen Befehlsnamen.

b) Der Befehl B

Dieser Befehl dient zum Ausdrucken von Beschreibungen aller Befehle und ihrer Spezifikationen, z. B.:

B,B'ERS'NUM,SPEZ	liefert eine Auflistung der Spezifikationsnamen für die Befehle B, ERS und NUM.
B,B,KURZ	liefert eine Kurzbeschreibung des Befehls B
B,BEISPIELE(B)	druckt Beispiele für den Befehl B aus.
B,MODUS(INF)	liefert eine Beschreibung der Spezifikation MODUS des Befehles INF
B,BEF,KURZ	liefert eine Liste aller Befehle mit ihren Spezifikationen wie sie unter VII angegeben ist.

Der Befehl B hat also dieselben Funktionen für AUFBEREITE-Befehle wie das INFORMIERE-Kommando des TR440 für TR440-Kommandos. Befehlsnamen und Spezifikationsnamen können ebenso wie dort im Rahmen der Eindeutigkeit durch einen Punkt abgekürzt werden.

c) Nähere Erläuterungen in Fehlersituationen:
Wird ein fehlerhafter Befehlsname oder Spezifikationsname eingegeben, so kann man sich bei der Korrekturanforderung von AUFBEREITE durch Angabe eines "*" (Multiplikationszeichen) wie bei Korrekturanfragen des Entschlüßlers des TR440 nähere Erklärungen zur Fehlerursache ausgeben lassen.

4.) Veränderung von mehreren Sätzen

Oftmals steht man vor dem Problem, in mehreren Sätzen bestimmte Zeichenfolgen durch bestimmte andere ersetzen oder sie löschen zu wollen.

Da es sehr zeitaufwendig und mühsam ist, dies in einzelnen Korrekturschritten satzweise tun zu müssen, hat AUFBEREITE die Möglichkeit, bestimmte Korrekturvorschriften zu speichern, und sie auf angebbare Sätze auf Wunsch anzuwenden.

Weil Dateibearbeitungen zwangsläufig ziemlich rechenzeitaufwendig sind, gibt es außerdem die Möglichkeit, beliebig viele solcher Korrekturvorschriften "in einem Durchgang" durch die Satzfolgen gleichzeitig durchzuführen. Das Vorgehen aus Benutzersicht ist dabei dreistufig:

a) Definieren der Korrekturvorschriften,
b) Aktivieren einiger oder aller definierten Korrekturvorschriften,
c) Anwenden aller aktiven Korrekturvorschriften.

Diese Dreistufigkeit hat einige entscheidende Vorteile:

Man kann sich einen Satz von Korrekturvorschriften definieren, die aber nicht immer alle gleichzeitig benötigt werden, sondern nur bei Bedarf "abgerufen", also aktiviert werden. Durch die Definitionsmöglichkeit müssen sie dann nicht immer wieder neu eingegeben werden.

Außerdem erlaubt es die Mehrstufigkeit, die aktiven Korrekturvorschriften in allen dateiverändernden Befehlen durchzuführen, also sogar während des Umnumerierens, des Kopierens aus anderen Dateien etc., was viel Rechenzeit ersparen kann.

Nun zur Benutzung:

a) Das Definieren von Ersetzungen genannten Korrekturvorschriften geschieht durch den Befehl DEF.

Jede Ersetzung wird durch einen "internen Namen", eine natürliche Zahl n (n ≤ maximal mögliche Anzahl der Ersetzungen), identifiziert und hat als Korrekturvorschrift einen Fremdstring, der nach folgendem Schema aufgebaut sein muß:

<Trennzeichen><String1><Trennzeichen><String2>

Das erste Zeichen des Fremdstrings definiert ein <Trennzeichen> , das durch sein zweites Auftreten innerhalb des Fremdstrings die Zeichenfolgen <String1> und <String2> voneinander trennt.

Die Bedeutung der Ersetzung ist im wesentlichen, daß die Zeichenfolge <String1> bei ihrem Auftreten durch die Zeichenfolge <String2> ersetzt werden soll.

Diese Ersetzung kann noch durch die Angabe eines Spaltenbereiches, in dem sie nur wirksam sein soll, eingeschränkt werden.

Beispiel: DEF,1,,/&MAX&ANTON

Die Ersetzung unter dem internen Namen 1 besagt: Überall, wo die Zeichenfolge "MAX" vorkommt, soll sie durch "ANTON" ersetzt werden. Als <Trennzeichen> wurde hierbei das "&" gewählt, da es in keinem der beiden Strings vorkommt.

DEF,2,10-20,/*A&B*C&D

Diese Ersetzung besagt: Überall in dem Spaltenbereich 10 bis 20 soll die Zeichenfolge "A&B" durch "C&D" ersetzt werden. Als Trennzeichen wurde hier das "*" gewählt.

Diese Ersetzung wirkt also z. B. nicht, wenn die Zeichenfolge "A&B" in Spalte 19 beginnt, da sie dann nicht komplett in dem angegebenen Spaltenbereich 10 bis 20 liegt.

DEF,3,-5,/?␣?

Ersetzung 3 bedeutet: Immer wenn am Satzanfang bis Spalte 5 einschließlich ein Blank vorkommt, soll dieses gelöscht werden, denn die Zeichenfolge <String2>, also die einzusetzende Zeichenfolge, ist ja leer.

b) Das Aktivieren von definierten Ersetzungen erfolgt mit dem Befehl ERS. Dabei werden zunächst alle definierten Ersetzungen inaktiv, und danach werden die angegebenen in ihrer Reihenfolge aktiviert.

Beispiel: ERS,1'2'4 Die Ersetzungen 1,2 und 4 werden aktiviert.

ERS,5'2'4'1 Die Reihenfolge der aktiven Ersetzungen ist nun: 5,2,4 und 1

Die Reihenfolge, in der die aktiven Ersetzungen später durchgeführt werden, ist nur abhängig von der Reihenfolge, in der sie beim ERS-Befehl aktiviert werden. Die Wichtigkeit der Reihenfolge läßt sich an fongendem Beispiel ablesen:

```
DEF,1,,/&MAX&KARL
DEF,2,,/&A&AA
ERS,1'2
```

Diese 3 Befehle bewirken, daß jedes A verdoppelt wird, aber durch die Aktivierungsreihenfolge wird festgelegt, daß ein A, das innerhalb der Zeichenfolge "MAX" auftaucht, nicht verdoppelt wird, sondern daß die Zeichenfolge "MAX" durch "KARL" ersetzt wird.

Daraus läßt sich schon erkennen, daß eine Zeichenfolge, die durch eine Ersetzung entstanden ist, von einer zweiten Ersetzung nicht mehr verändert werden kann, denn sonst würde das "A" in "KARL" ja durch Ersetzung 2 nochmal verdoppelt, was nicht beabsichtigt war. Wollte man die Zeichenfolge "MAX" nur bei ihrem ersten Auftreten im Satz in "KARL" wandeln, so kann man dies erreichen durch Angabe einer Anzahl im ERS-Befehl.

Beispiel: ERS,2,,1 bewirkt, daß die Ersetzung 2 nur einmal pro Satz durchgeführt wird.

Sollen alle Ersetzungen aktiviert werden, so läßt sich das leicht durch ERS, -STD- erreichen. Die Reihenfolge ist dann aufsteigend nach den internen Namen.

Wenn also die Ersetzungen 2,5,10 und 12 definiert sind, so bewirkt

```
ERS, -STD-
```

dasselbe wie

```
ERS,2'5'10'12.
```

c) Passiviert werden immer alle Ersetzungen, die beim ERS-Befehl nicht mit angegeben werden.

Durch den Befehl ERS ohne Spezifikationen werden deshalb alle Ersetzungen inaktiv.

d) Das Durchführen von aktiven Ersetzungen geschieht wie gesagt bei jedem dateiverändernden Befehl.

Im Normalfall wird mit dem Befehl E gearbeitet, der die spezifizierten Sätze der Reihe nach liest, aktive Ersetzungen durchführt und den jeweiligen Satz, falls er verändert wurde, zurückschreibt.

Beispiele: E, 1- führe alle aktiven Ersetzungen für alle Sätze der Datei durch.

E,100-1000'10000-20000,,/EGON

führe alle aktiven Ersetzungen in allen Sätzen der Bereiche 100 bis 1000 und 10000 bis 20000 durch, die die Zeichenfolge "EGON" enthalten.

Die Suchbedingung für Sätze, in denen Ersetzungen durchgeführt werden sollen, ist also völlig unabhängig von den tatsächlichen aktiven Ersetzungen.

Nun noch einige Beispiele, die das Zusammenspiel der 3 Stufen im Zusammenhang zeigen:

α) In allen Sätzen der Datei im Bereich 1000 bis 3000, die in Spalte 1 bis 5 nur Leerzeichen entfalten (z. B. kein FORTRAN-LABEL), soll die Zeichenfolge "Z(I)" durch "NZ(I)" ersetzt werden:

```
DEF,1,,/&Z(I)&NZ(I)
ERS,1
E,1000-3000,1-5,/␣␣␣␣␣
```

β) Es handele sich um eine ALGOL60-Quelle, in der bestimmte Delimiter in die spezielle TR440-Ersatzdarstellung umgewandelt werden sollen:

```
DEF,1,,/&.=&:=
DEF,2,,/&..&:
DEF,3,,/&(/&[
DEF,4,,/&/)&]
ERS,1'2'3'4,NP
E,1-
```

Dadurch werden alle vier Delimiteränderungen "in einem Durchlauf" durch die Quelle durchgeführt, so daß jeder Satz nur einmal gelesen und nur zurückgeschrieben wird, wenn er verändert wurde. Die Angabe von NP beim ERS-Befehl bewirkt, daß auch nur die veränderten Sätze protokolliert werden.

5.) Eine spezielle Möglichkeit einer Korrekturvorschrift, die unabhängig von den Ersetzungen wirkt, besteht aus dem Entfernen von Leerzeichen aus Sätzen.

Dazu dient der Befehl TRIM.
Er ermöglicht es, zu steuern, ob Blanks am Satzanfang, am Satzende oder ganz zu entfernen sind:

TRIM,V bewirkt ein Entfernen vorne im Satz
TRIM,H ein Entfernen hinten im Satz,
TRIM,V'H vorne und hinten im Satz,
TRIM,A ein Entfernen aller Blanks im Satz.

TRIM ohne Spezifikationswert macht eine solche Einstellung wieder rückgängig, so daß keine Blanks entfernt werden.

Diese Einstellung mit dem Befehl TRIM bewirkt gewissermaßen die Definition und Aktivierung einer speziellen, besonders schnellen Ersetzung.

Sind zusätzlich andere Ersetzungen definiert und aktiv, so wirkt die TRIM-Einstellung vor den aktiven Ersetzungen.

Durch den TRIM-Befehl allein werden genau wie bei den Ersetzungen noch keine Sätze verändert, sondern die Blanks werden erst bei irgendwelchen dateiverändernden Befehlen entfernt.

6.) Protokollsteuerungen

Es gibt zwei verschiedene Möglichkeiten der Beeinflussung von Protokollierungen:

Einmal eine globale Steuerung, die voreingestellt ist und bewirkt, daß bei allen Dateiveränderungen die betroffenen Sätze protokolliert werden:

PROT,A'KO bewirkt, daß alle Dateiveränderungen protokolliert werden und zwar zusätzlich auf der Konsole.

PROT,- bewirkt, daß nichts protokolliert wird.

Die zweite Möglichkeit ist, die abweichend vom global eingestellten Protokollierungsmodus für einen einzelnen Befehl die Protokollierung ein- oder auszuschalten, indem man dem entsprechenden Befehl ein "P" bzw. ein "N" voranstellt. Wenn also z. B. das Protokoll global eingeschaltet ist (Voreinstellung), so kann man durch den Befehl NL statt L Sätze löschen, ohne daß die gelöschten Sätze protokolliert werden.

Entsprechend kann man bei global ausgeschaltetem Protokoll z. B. durch den Befehl PE statt E erreichen, daß die durch Ersetzungen veränderten Sätze protokolliert werden. Eine solche Protokollsteuerung durch Voranstellen des Buchstabens "N" oder "P" wirkt jeweils nur für den einen Befehl.

Bei allen Befehlen, die ohne diesen führenden Buchstaben eingegeben werden, wird die aktuell eingestellte globale Protokollsteuerung wirksam.

Ist bei einem dateiverändernden Befehl das globale oder das explizite Protokoll eingeschaltet, so wird jede Veränderung der Datei protokolliert, wobei durch einen der Satznummer vorangestellten Buchstaben gekennzeichnet wird, welcher Art die Veränderung ist. Dabei bedeuten:

- K der Satz wurde korrigiert
- L der Satz wurde gelöscht
- D der Satz entstand durch Duplizierung
- N der Satz entstand durch Umnumerierung
- P der Satz wird nur protokolliert, wurde also nicht verändert.
- E der Satz ersetzt, also überschreibt einen bereits vorher existierenden Satz.
- V dieser Satz stand vorher in der Datei, wird aber durch den folgenden protokollierten ersetzt (nur, wenn V als Teilwert des globalen Protokolls eingestellt war).

7.) Sicherung gegen versehentliche Zerstörungen

Es stehen zwei Möglichkeiten zur Verfügung, mit Hilfe derer der Benutzer sich vor einer versehentlichen Zerstörung seiner Datei schützen kann.

a) Er kann einen Dateibereich festlegen, in dem ausschließlich gearbeitet werden soll. Zum Beispiel:

BER,100000-200000

Dieser Befehl bewirkt

α) daß nur in dem Dateibereich von Satz 100000 bis Satz 200000 auf die Datei zugegriffen werden kann.

β) Zusätzlich werden alle Sätze mit der Anfangssatznummer translatiert, also auf den Satz 105300 wird z. B. mit der Nummer 5300 zugegriffen.

b) Nach dem Befehl SYSI, -STD- "merkt sich" der Operator AUFBEREITE alle Änderungen, die in der Datei durchgeführt werden und ist somit in der Lage, jederzeit den Originalzustand der Datei wiederherstellen zu können, wie er zu Beginn des AUFBEREITE-Laufes war.

Dieses Wiederherstellen des Originalzustandes geschieht durch den Befehl SYSI, GET. Der Effekt ist also derselbe, als wenn zu Beginn eine komplette Kopie der Datei in einer Scratchdatei angelegt würde, aber wesentlich weniger platzaufwendig, da nur die veränderten Sätze kopiert werden.

Auch wenn ein Satz mehrmals verändert wurde, wird immer der Originalzustand vor der ersten Veränderung vermerkt.

Will man zwischendurch den aktuellen Stand der Datei zum "Originalzustand" machen, da alle Änderungen ok sind, kann man dies durch SYSI, NORM ohne Platz- oder Rechenzeit-Aufwand erreichen, da dabei lediglich alle Vermerke gelöscht werden müssen.

Sollen nur bestimmte Sätze wieder in ihren Originalzustand gebracht werden, so läßt sich das durch

SYSI,GET,<Satzbereich> erreichen.

Will man bei der Sicherung des Originalzustandes auch gegen Systemzusammenbrüche gefeit sein, so kann man als "Sicherungsdatei", also als Datei, in der alle Vermerke über Änderungen abgelegt werden, dem Operator auch eine LF- oder WSP-Datei anbieten, die zum Schreiben eingeschleust sein muß.

Das geschieht durch

SYSI,-STD-,<Dateiname>

Diese Datei wird jedoch nicht automatisch reserviert oder bereinigt.

Standardmäßig benutzt AUFBEREITE die Datei &ZWSICHERUNG(9999.99), die bei Bedarf automatisch kreiert wird.

8.) Vorgehen bei Platzmangel in der Datei

Wird eine Datei zu klein, so versucht der Operator zunächst, sie durch Reservierung zu vergrößern.Gelingt dies nicht, so fragt er im Dialog an:

Datei bereinigen (J/N/WDH) ¤:

mögliche Reaktionen darauf:

a) "J":
Die Datei wird bereinigt und danach wird an der Unterbrechungsstelle fortgefahren.

b) "N":
Der augenblickliche Vorgang wird abgebrochen, und der Operator kehrt in die Grundstufe zurück.
Alle nicht interpretierten Befehle werden protokolliert.

c) "WDH":
Der letzte Schreibvorgang, bei dem der Fehler aufgetreten war "Datei voll", wird wiederholt. Das ist z. B. sinnvoll, wenn der Benutzer durch vorrangige TR440-Kommandos Platz zur weiteren Informationsaufnahme in der Datei geschaffen hat.

9.) Behandlung von syntaktischen Fehlern in Befehlen

Wird bei einem Befehl ein syntaktisch fehlerhafter Spezifikationswert angegeben, so fragt der Operator AUFBEREITE im Dialog den richtigen Spezifikationswert an. Das hat den Vorteil, daß nicht der gesamte Befehl und alle folgenden Befehle noch einmal eingegeben werden müssen, sondern nur der fehlerhafte oder fehlende Wert. Eine solche Korrekturanforderung sieht z. B. folgendermaßen aus:

a) Befehl ␣:K,MIST␣.

+++++ K: ← Befehlsname

MIST unzul., Bereich=␣: ← Spezifikationsname

fehlerhafter Spezifikationswert

b) Befehl ␣:K␣.

++++++K:

Bereich fehlt, Bereich = ␣:

Auf eine Korrekturanforderung sind folgende Eingaben möglich:

α) Eingabe des korrekten Spezifikationswertes.

Der Befehl wird mit dem korrekten Wert ausgeführt, indem dieser anstelle des fehlerhaften oder fehlenden Spezifikationswertes eingesetzt wird.

β) Leere Eingabe.

Der Befehl wird nicht ausgeführt, sondern übergangen.

γ) Eingabe von BEENDE

Der Befehl wird übergangen, und die restliche , noch nicht interpretierte Eingabe unter Protokollierung ignoriert - danach werden wieder Befehle erfragt.

δ) Eingabe eines "*":

Es wird eine Kurzbeschreibung des Befehls und des Spezifikationswertes ausgegeben, bei dem der Fehler auftrat - danach wird die Anfrage nach dem korrekten Spezifikationswert wiederholt.

ε) Einfügung vorrangiger Befehle durch VORRANG,/ [<Befehl>]$^{\infty}$␣.

Der aktuell anstehende, fehlerhafte Befehl wird ignoriert, und statt seiner werden die angegebenen Befehle vorrangig durchgeführt, danach wird an der Unterbrechungsstelle fortgefahren.

10.) Kommentare in Befehlsfolgen

In der Befehlsstufe werden Zeilen, die mit einem Gleichheitszeichen beginnen, als Kommentare aufgefaßt und nicht interpretiert.

Beispiel: Befehl ␣: = Dies ist ein Kommentar␣.

Befehl ␣:

STUFE 3

1. Die Bearbeitung von SEQ-Dateien ergibt sich ganz zwangsläufig, da die verwendeten E/A-Routinen (BODAT) das eigenständig bewerkstelligen.

   Das bedeutet: Lesend (z. B. beim Protokollieren) können die Satznummern genauso angegeben werden wie bei RAN- oder RAM-Dateien, also auch mit mehreren Teil-Bereichen, die nicht aufsteigend sortiert sein müssen.

   P, 100-200'10000+5' 5+10

   ist zum Beispiel auch für SEQ-Dateien erlaubt.

   Dies wird dadurch erreicht, daß intern (evtl. nach einem REWIND oder FORWIND) alle Sätze rückwärts oder vorwärts (je nachdem, welche Richtung schneller ist) gelesen werden - bis zur verlangten Satznummer.

   Beim Schreiben ist zu beachten, daß hinter einem geschriebenen Satz alle nachfolgenden Sätze gelöscht sind. Aus diesem Grund ist auch der Korrigierbefehl verboten und einige andere.

   Soll eine Satznummer geschrieben werden, die größer als die größte vorhandene ist, so wird direkt hinter die höchste geschrieben. Dadurch ist es z. B. möglich, zwei SEQ-Dateien zu verknüpfen (Befehl GET) oder eine SEQ-Datei umzunumerieren, wenn der letzte Satz der Datei mit in dem umzunumerierenden Bereich liegt (besonders sinnvoll zum Vertauschen von Bereichen).

2.) Bei der Bearbeitung von A-Dateien, also Dateien mit "Ausgabezeichen" als Satzelemente, ist eigentlich nur zu beachten, daß das erste Zeichen jedes Satzes ein Vorschubsteuerzeichen sein muß, und daß am Satzende eine Folge von 0 bis 5 Ignores entstehen kann, da A-Dateien keinen Oktadenzähler im letzten Ganzwort enthalten. Bei A-Dateien macht sich die Möglichkeit besonders bezahlt, das Zeichen für das Zeilenende frei wählen zu können (Befehl ZENDE),da dadurch auch das Vorschubsteuerzeichen 21 ("Newline") benutzt werden kann, das sonst immer die Funktion des Zeilentrennens hat. Der Operator AUFBEREITE prüft jedoch nicht ab, ob das erste Zeichen eines Satzes tatsächlich ein gültiges Vorschubsteuerzeichen ist.

   Das einfachste Verfahren, Sätze mit Vorschubsteuerzeichen zu erzeugen (insbesondere mit "Newline") ist, sich eine entsprechende Ersetzung zu definieren, wobei man während der Definition dieser Ersetzung kurzfristig das ZENDE-Zeichen umstellt.

   Beispiel:

siehe nächste Seite

Beispiel:

```
Befehl ¤: ZENDE,?          <- Hier wirkt noch einmal "Newline" als Zeilentrenner
DEF,4,1,/.&.¤021 ?ZENDE,21?ERS,4
S,1000+10,/                <- Hier wirkt das "?" als Zeilentrenner
&SATZ1
&SATZ2
&SATZ3¤.
```

Dadurch wird jedes "&" in Spalte 1 durch "Newline" mit dem Zentralcodewert 21 ersetzt bei der Eingabe der 3 Sätze 1000,1010 und 1020 und allen folgenden dateiverändernden Befehlen.

3.) Sehr lange Sätze lassen sich bequem mit AUFBEREITE bearbeiten.

Voraussetzung ist natürlich, daß genügend Kernspeicher zur Verfügung steht, und daß entweder im Kommando AUFBEREITE die Spezifikation SATZLAENGE entsprechend besetzt wurde oder derselbe Effekt durch den Befehl SATZLG zwischendurch im Lauf erzielt wurde.

Für Sätze mit 1000 Zeichen oder mehr muß man jedoch entweder im Modus FSR oder mit dem Befehl FSRK korrigieren, da die Größe der Bildschirme nicht ausreicht, so lange Sätze darzustellen und deshalb die Sichtfenstermethode zum Korrigieren ungeeignet ist. Bei der fernschreiberanalogen Korrektur gibt es nämlich die Möglichkeit, sich nur bestimmte Ausschnitte aus den Sätzen zur Korrektur vorlegen zu lassen.

Einen solchen zur Korrektur vorgesehenen Spaltenbereich innerhalb langer Sätze wählt man global mit dem Befehl KORBER aus.

4.) Auch nichtdarstellbare Zeichen in den Sätzen lassen sich bequem mit AUFBEREITE bearbeiten.

Das liegt an folgenden Leistungen des Operators:

a) Bei der Einstellung WANDEL, -STD- (voreingestellt) werden alle nicht auf dem Arbeitsterminal darstellbaren Zeichen in ihre dezimale Ersatzdarstellung ¤<Ziffer><Ziffer><Ziffer> gewandelt, so daß sie erstens leicht lesbar und zweitens bei der Sichtfenstermethode des Korrigierens auch wieder richtig einlesbar sind. Lediglich bei der fernschreiberanalogen Korrektur werden sie als Ausrufezeichen vorgelegt, um ein spaltengerechtes Arbeiten zu gewährleisten.

b) Für sehr schlecht handhabbare Zeichen wie "Newline" oder gar überhaupt nicht eingebbare Zeichen wie "EM" mit dem Zentralcodewert 33 sowie für andere Spezialfälle, in denen es nicht möglich ist, Zeichen in der Form ¤<Ziffer><Ziffer><Ziffer>

einzugeben, wurde eine Möglichkeit eingebaut, Ersetzungen in sedezimaler Form anzugeben, wenn nämlich der einzusetzende String der Ersetzung wiederum mit dem ⟨Trenner⟩ der Ersetzung beginnt.

Beispiel: DEF,1,,/?%??AF2100AF

Hierbei wird eine Ersetzung definiert, die jedes Prozentzeichen in die Zeichenfolge Blank-EM-Ignore-Blank wandelt.

Man muß darauf achten, daß eine gerade Anzahl von Tetraden (Sedezimalziffern) angegeben wird.

Formal lautet diese Form der Ersetzung:

$$/\langle\text{Trenner}\rangle\left[\langle\text{Oktade}\rangle\right]_0^{\infty}\langle\text{Trenner}\rangle\left[\langle\text{Tetrade}\rangle\langle\text{Tetrade}\rangle\right]_1^{\infty}$$

5.) Dadurch, daß die aktiven definierten Ersetzungen bei allen dateiverändernden Befehlen wirksam sind, also auch bei direkten Eingabebefehlen wie S und EIN, läßt sich AUFBEREITE auch als Eingabemakroprozessor benutzen.

Beispiel: In ALGOL60 gibt es viele unangenehm lange Delimiter wie 'PROCEDURE' etc.

ALGOL60-Quellen lassen sich aber sehr bequem eintragen, wenn man sich folgenden Kommandos bedient:

```
¤AUFBEREITE,DAT,DEF.=20,INF.=/
DEF,1,,/?&P?'PROCEDURE'
DEF,2,,/?&I?'INTEGER'
DEF,3,,/?&(?'BEGIN'
DEF,4,,/?&)?'END'
DEF,5,,/?&B?'BOOLEAN'
DEF,6,,/?&R?'REAL'
DEF,7,,/?&F?'FOR'
ERS,1'2'3'4'5'6'7¤.
```

Wenn man jetzt im AUFBEREITE (evtl. sogar weiter im INFORMATIONS-Fremdstring) mit dem Befehl S oder EIN Sätze einträgt, so werden die Sätze entsprechend den Definitionen quasi auf dem Weg zwischen Terminal und Datei expandiert.

Wird dieses Kommando einschließlich Fremdstring noch in eine Kommandoprozedur verpackt, so macht das Eintragen von ALGOL60-Quellen fast Spaß.

6.) Das Arbeiten an irgendwelchen Terminals mit Blockmodus (Sichtfenstermethode) mit dem Befehl K. Standardmäßig vorgesehen als Blockmodus - Terminals sind:

das SIG100 mit Sichtgerätevermittler SIV100,
das SIG50,
das SIG51 und
das Infotongerät VISTAR.

Die Sichtfenstermethode beruht, allgemein gesagt darauf, daß ein Teil der Datei auf dem Bildschirm zur Korrektur vorgelegt wird (Sichtfenster). Der Bildschirminhalt wird dann mittels Hardwarefunktionen des Gerätes wie "Insert Charakter", "Delete Charakter", "Insert Line", "Delete Line" sowie Überschreiben von Zeichen etc. geändert und als Eingabe zum Rechner zurückgeschickt. Der Operator AUFBEREITE untersucht den wieder eingelesenen Bildschirminhalt auf Veränderungen und führt die festgestellten Änderungen auch in der Datei durch.

Jede Zeile hat dabei den Aufbau:

<Satznummer><Blank><Satzinhalt>.

In dieser an sich trivialen Benutzung hier noch einige gerätespezifischen Hinweise:

α) Beim SIG100 muß man darauf achten, daß jede Zeile wirklich mit einem Zeilenvorschub abgeschlossen wird, was man leicht vergessen kann. Außerdem muß die Eingabe im "Fernschreibmodus" und nicht im "Textmodus" oder "Graphikmodus" abgeschickt werden. Damit kann man sich aber im Normalfall nur vertun, wenn zwischendurch durch ein

"BUSAS *EINGABE GELOESCHT"

die Rechnerausgabe im "Textmodus" wiederholt wird.

Wird nicht im "Fernschreibmodus" eingelesen, so meldet AUFBEREITE dies und schreibt den gesamten Bildschirminhalt, der eingelesen wurde, wieder aus, stellt den "Fernschreibmodus" ein, und ein einfaches erneutes Abschicken des Bildschirminhaltes genügt zur richtigen Abarbeitung.

β) Beim SIG50 und SIG51 ist zu beachten, daß die "Beginn Abschickbereich"- und "Ende Abschickbereich"-Klammern richtig gesetzt sind.

Beim VISTAR muß man darauf achten, daß wirklich der Blockmodus eingestellt ist. Abgeschickt wird der Bildschirminhalt durch gleichzeitiges Drücken von "SHIFT" und "XMIT PAGE".

Enthält die Datei Kleinbuchstaben, so muß unbedingt der Code SC4 eingestellt werden durch

¤XUM,COD=SC4¤.

da im Normalfall der Code SC4G voreingestellt ist, der den TR86 dazu veranlaßt, bei der Eingabe alle Kleinbuchstaben in Großbuchstaben umzukodieren.

Beim Blockmodus-Korrigieren am VISTAR wird die Möglichkeit, mit dem Befehl ZENDE das Zeilenendezeichen umzustellen, besonders wichtig, da das VISTAR die Eigenschaft hat, im Blockmodus das normale Zeilenende nicht mit zu übertragen. Als Ersatz dafür schreibt der Operator AUFBEREITE hinter jede Zeile das eingestellte ZENDE-Zeichen mit aus und wertet es bei der Eingabe entsprechend als Zeilenende aus.

Voreingestellt ist das Zeichen "Underscore" mit dem Zentralcodewert 110, das am VISTAR durch " ← " dargestellt wird.

Auch wenn neue Zeilen dem Bildschirminhalt hinzugefügt werden, muß der Benutzer jeden Satz mit dem eingestellten ZENDE-Zeichen abschließen. Das Leerzeichen, das im Blockmodus vom Gerät hinter jede Eingabezeile angehängt wird, wird automatisch vom Operator entfernt. Will man zwischendurch mal im "Charaktermodus" am VISTAR arbeiten, so ist zu beachten, daß das normale "Newline" nicht als Zeilentrennungszeichen interpretiert wird, also z. B. keine Befehle voneinander trennt.

Um diese Schwierigkeit, die eigentlich gar keine ist, zu umgehen, muß man im "Charaktermodus" die gesamte Eingabe ohne expliziten Zeilenvorschub eingeben und die einzelnen "Zeilen" nur durch das ZENDE-Zeichen innerhalb einer echten Eingabezeile trennen.

Wenn Ersatzdarstellungen ¤<Ziffer><Ziffer><Ziffer> auch mit einem Zentralcoedewert < 64 möglich sind, kann man mit ZENDE,21 auch das Zeichen "Newline" als ZENDE-Zeichen wählen und damit die eben genannte "Schwierigkeit" umgehen.

Implizit wird als Gerät "VISTAR" erkannt, wenn die eingestellte Zeilenbreite des Terminals 240 ist. Ist das nicht der Fall, so muß entweder im Kommando AUFBEREITE MODUS=VISTAR angegeben werden oder man muß zwischendurch den Befehl MODUS,VISTAR geben.

γ) Arbeitet man an einem Terminal, das implizit als ein Blockmodus-Gerät erkannt wird, man möchte aber mit der fernschreiberanalogen Methode korrigieren, so kann man das auf zwei Arten erreichen:

Entweder man korrigiert mit dem Befehl FSRK oder man stellt explizit den FSR-Modus ein durch Angabe von MODUS=FSR im Kommando AUFBEREITE oder durch den Befehl MODUS, FSR.

δ) Es gibt eine Reihe von Terminals mit der Möglichkeit des Blockmodus, die irgendwelche abstrusen Konventionen haben.

Die meisten dieser Terminals lassen sich auch im VISTAR-Modus betreiben. Dazu muß man nur (durch Ausprobieren) wissen, welches Zeichen als Zeilenendezeichen abgeschickt wird, und durch einen entsprechenden ZENDE-Befehl dieses Zeichen als ZENDE-Zeichen einstellen. Wird vom Gerät selbst automatisch ein solches Zeilenende-Zeichen generiert, so muß man zusätzlich verhindern, daß dieses Zeichen von AUFBEREITE noch einmal an jeden Satz angehängt wird, indem man es in ein Ignore umcodieren läßt.

Zum Beispiel gibt es Geräte, die im Blockmodus nach jeder Zeile als Zeilenendezeichen das Zeichen mit dem Zentralcodewert 60 abschicken.

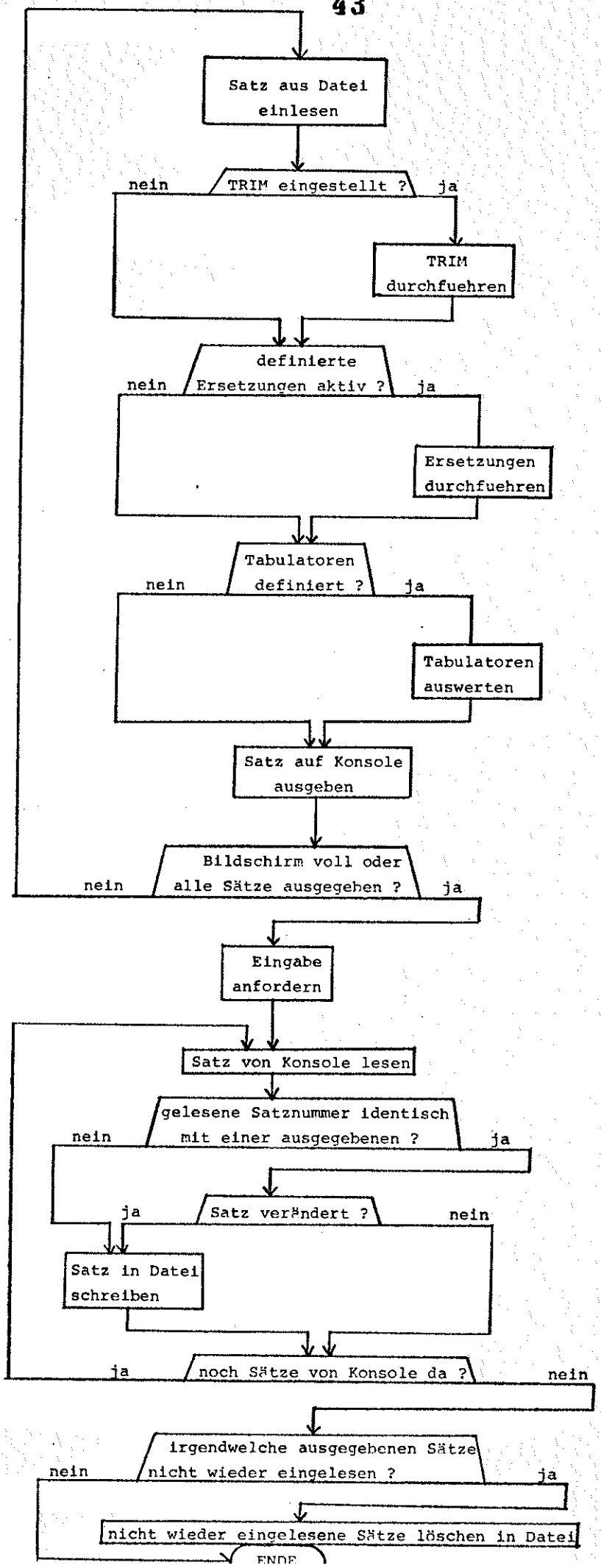
An diesen Geräten kann man im Blockmodus arbeiten, wenn man zu Operatorlaufbeginn folgende Befehle eingibt:

MODUS,VISTAR	(Modus einstellen)
ZENDE,60	(ZENDE-Zeichen wählen)
CODTAB,060:000	(ZENDE-Zeichen, da automatisch vom Gerät hinzugefügt, in Ignore umkodieren)
TRIM,H	(falls das Gerät Blanks hinter Zeilen anfügt)

Wenn man bei solchen Geräten die Wahlmöglichkeit hat, den Bildspeicher mit Blanks oder Ignores vorzulöschen, sollte man unbedingt Ignores wählen.

Wenn das Gerät nicht automatisch ein Zeilenende-Zeichen generiert, so muß man den CODTAB-Befehl weglassen.

ε) Korrigiert wird mit dem Befehl K.
Die Vorgehensweise des Operators ist dabei folgende:


Satz aus Datei einlesen
TRIM eingestellt ?
nein
ja
TRIM durchfuehren
definierte Ersetzungen aktiv ?
nein
ja
Ersetzungen durchfuehren
Tabulatoren definiert ?
nein
ja
Tabulatoren auswerten
Satz auf Konsole ausgeben
Bildschirm voll oder alle Sätze ausgegeben ?
nein
ja
Eingabe anfordern
Satz von Konsole lesen
gelesene Satznummer identisch mit einer ausgegebenen ?
nein
ja
Satz verändert ?
ja
nein
Satz in Datei schreiben
noch Sätze von Konsole da ?
ja
nein
irgendwelche ausgegebenen Sätze nicht wieder eingelesen ?
nein
ja
nicht wieder eingelesene Sätze löschen in Datei
ENDE

7. Andere Befehls-Eingabemedien als die Konsole:

a) Zur Vereinfachung der Handhabung gibt es zunächst die Möglichkeit, dem Operator bereits beim Start als Fremdstring Befehle mitzugeben, z. B. durch eine Kommandoprozedur etc.

Das geschieht beim Kommando AUFBEREITE durch die Spezifikation INFORMATION und beim STARTE-Kommando durch die DATEN-Spezifikation.

Die reinen dialogorientierten Befehle wie K, FSRK etc. sind im Fremdstring natürlich verboten.

Durch diese Möglichkeit ist AUFBEREITE ebenso wie im Dialog im Abschnitt benutzbar.

Wenn im Gespräch der Fremdstring abgearbeitet ist, wird automatisch im Dialogmodus mit Eingabeanforderung fortgefahren.

b) Als zweite wesentliche Erweiterung der Möglichkeiten von AUFBEREITE kann man auch Befehle ausführen lassen, die in Dateien stehen, und zwar mit dem Befehl TUE .

Die auszuführenden Befehle können sowohl in der aktuell bearbeiteten Datei selbst als auch in irgendwelchen anderen Dateien stehen.

Dabei können durchaus mehrere Befehle in einem Satz stehen, wenn sie durch das aktuelle ZENDE-Zeichen getrennt sind. Zwischen die Befehle aus verschiedenen Sätzen fügt AUFBEREITE selber das aktuelle ZENDE-Zeichen (z. B. "Newline") ein. Dadurch ergibt sich z. B. die Möglichkeit, eine Menge von Befehlen mehrmals zu durchlaufen, ohne sie mehrmals eingeben zu müssen.

Beispiel: α) In der aktuellen Datei mögen im Bereich von 1000 bis 1100 eine Reihe von Befehlen stehen, die 3-mal durchlaufen werden sollen:

```
TUE,,1000-1100'1000-1100'1000-1100
```

β) In der Datei TEST in Datenbasis DABA mögen von Satz 900000 ab Befehle stehen, die ausgeführt werden sollen:

```
TUE,DABA.TEST,900000-
```

γ) In der aktuell bearbeiteten Datei mögen ab Satz 900000 Befehle stehen, die durch eine Art Inhaltsverzeichnis sortiert sind nach folgendem Schema:

ein Satz mit einer kennzeichnenden Zeichenfolge
und danach 5 Sätze mit Befehlen.

Die Befehlsfolge, die durch die Zeichenfolge "ALG60" gekennzeichnet ist, soll ausgeführt werden:

```
SUCH,900000+1,,/ALG60
SUCH,*A+2
TUE,,*A+5
```

Der Befehl TUE ist voll rekursiv benutzbar; in den Sätzen, die durch TUE als Befehle ausgeführt werden sollen, dürfen also wieder TUE-Befehle stehen etc. Es ist jedoch darauf zu achten, daß sämtliche durch TUE noch auszuführenden Befehle zusammen nicht mehr als 120000 Zeichen umfassen, was aber wohl keine wesentliche Einschränkung ist (die bereits interpretierten Befehle vor TUE werden natürlich nicht mitgezählt).

Die Vorgehensweise beim Befehl TUE ist eigentlich trivial: Alle durch den Befehl spezifizierten Sätze werden sofort beim TUE-Befehl in den Konsol-Eingabepuffer eingelesen und ersetzen dort den TUE-Befehl.

Etwa schon im Eingabepuffer stehende Befehle (hinter dem TUE-Befehl) werden, durch das aktuelle ZENDE-Zeichen getrennt, dahinter gekettet.

8.) Handhabung von komplizierten Suchbedingungen und Ersetzungsvorschriften

Bisher haben wir nur die direkt in dem jeweiligen Befehl angegebene Suchbedingung kennengelernt, die aus

<Spaltenbereich>,<Suchstring> oder Komibinationen

dieser beiden Formen besteht.

a) Darüber hinaus gibt es die Möglichkeit, eine globale Suchbedingung zu definieren, und zwar mit Hilfe des Befehles KRIT.

Zunächst ist es möglich, eine direkt angegebene Suchbedingung zu negieren in dem Sinne:

Protokolliere alle Sätze, in denen die Zeichenfolge "XY" nicht vorkommt

```
KRIT,-
P,1-,,/XY
```

oder:

Protokolliere alle Sätze, in denen das erste Zeichen kein "Z" ist:

```
KRIT,-
P,1-,1,/Z
```

Durch KRIT,- werden also alle folgenden Suchbedingungen negiert. Soll die Negierung wieder rückgängig gemacht werden, so gibt man den Befehl KRIT, -STD- (dieser Zustand ist auch zu Beginn voreingestellt).

b) Eine weitere Möglichkeit besteht darin, mit Hilfe der Spezifikation GLOBALKRITERIUM des Befehles KRIT eine ausgedehnte matching-Bedingung zu spezifizieren. Dabei wird der zu suchende Satz in Form einer Art "Maske" beschrieben.

Umgangssprachlich ausgedrückt heißt eine solche match-Bedingung z. B.:

> Suche alle Sätze, in denen zunächst die Zeichenfolge "ANTON", dann eine beliebige, beliebig lange Zeichenfolge, dann die Zeichenfolge "BERTA", danach eine beliebige Anzahl von Blanks und danach die Zeichenfolge "CAESAR" vorkommt.

Die expliziten Zeichenfolgen "ANTON", "BERTA" und "CAESAR" werden dabei einfach hingeschrieben, und für die Bedingungen "beliebige, beliebig lange Zeichenfolge" und "beliebige Anzahl von Blanks" wird jeweils ein spezielles Zeichen als Stellvertreter an die jeweilige Stelle geschrieben.

Diese Stellvertreter nennen wir ‹Beliebigzeichen› und ‹Blankzeichen›.
Um völlig variabel in den Zeichenfolgen zu sein (das ‹Beliebigzeichen› und das ‹Blankzeichen› dürfen ja nicht in den expliziten Zeichenfolgen vorkommen), werden die beiden Spezialzeichen bei jedem GLOBALKRITERIUM neu definiert.

Unsere Suchbedingung könnte dann z. B. so angegeben werden:

```
KRIT,-STD-,/ω.ANTON.BERTA␣CAESAR
```

Das erste Zeichen des Fremdstrings unter GLOBALKRITERIUM (2. Spezifikation von KRIT) definiert jeweils das ‹Blankzeichen› und das zweite Zeichen des Fremdstrings das ‹Beliebigzeichen›. In unserem Beispiel wird also das Leerzeichen als ‹Blankzeichen› und der Punkt als ‹Beliebigzeichen› definiert. Man könnte z. B. auch schreiben:

```
KRIT,-STD-,/%&ANTON&BERTA%CAESAR
```

Der Wert "-STD-" muß unter der 1. Spezifikation MODUS angegeben werden, damit die Suchbedingung nicht negiert wird (s.o.).

Weitere Beispiele:

α) Suche in der ALGOL60-Quelle jedes Auftreten eines Elementes des Arrays "A1":

```
KRIT,-STD-,/ω&A1[&]
P,1-
```

β) Suche die Zeichenfolge "XYZ", unabhängig davon, ob sie vielleicht gesperrt geschrieben ist:

```
KRIT,-STD-,/␣&X␣Y␣Z
P,1-
```

Ist bei dem jeweiligen Befehl zusätzlich zu dem GLOBALKRITERIUM noch eine explizite Suchbedingung angegeben, so werden diese beiden durch "und" verknüpft.

Beispiel: Suche in der (FORTRAN-)Quelle Statements mit einer arithmetischen IF-Anweisung und ohne Label, also Statements, die in den ersten 6 Spalten nur Blanks und dann die Zeichenfolge "IF(", dann einen beliebigen arithmetischen Ausdruck, dann eine "runde Klammer zu" und danach noch irgendwo zwei Kommata enthalten:

```
KRIT,-STD-,/␣.IF(.).,.,
P,1-,1-6,/␣␣␣␣␣␣
```

Wenn in der Zeichenfolge "IF(" noch Leerzeichen vorkommen können, müßte man schreiben:

```
KRIT,-STD-,/␣.I␣F␣(.).,.,
P,1-,1-6,/␣␣␣␣␣␣
```

Ein globales Kriterium wird wieder gelöscht, indem man einen KRIT-Befehl mit GLOBALKRITERIUM= "undefiniert" angibt, also z. B.:

```
KRIT,-STD-
```

c) Mit Hilfe des Befehles SUCH kann man auch Suchbedingungen spezifizieren, die sich über mehrere Sätze der Datei erstrecken.

Das geschieht über den internen Zeiger, der beim Befehl SUCH auf einen Satz gesetzt wird, und der durch die Angabe von "*A" als Satznummer angesprochen werden kann.

Beispiele: α) Es handle sich um eine ALGOL60-Quelle:

Protokolliere das erste 'BEGIN' der Prozedur PROZ:

```
SUCH,1-,,/'PROCEDURE'PROZ
P,*A+1,,/'BEGIN'
```

Durch den ersten Befehl wird dabei auf die Deklarationszeile der Prozedur PROZ "positioniert" und danach das erste Auftreten der Zeichenfolge "'BEGIN'" gesucht.

β) Es handle sich um eine FORTRAN-Quelle:

Korrigiere alle Rückwärtssprünge auf das Label 357 :

```
SUCH,1-,1-5,/357
SUCH,*A+2
K,*A-,,/357
```

Der erste Befehl "positioniert" auf die Deklarationszeile des Labels 357, der zweite auf eine Zeile dahinter, da die Deklarationszeile nicht mit korrigiert werden soll, und der dritte schließlich legt alle nachfolgenden Bezüge auf das Label 357 zur Korrektur vor.

γ) Lösche die dritte und vierte Zeile des Paragraphen 5:

```
SUCH,1-,,/PARAGRAPH 5
SUCH,*A+4
L,*A+2
```

d) Mit Hilfe des Befehles TUE lassen sich Ersetzungsvorschriften durchführen, die sich über mehrere Sätze erstrecken.

Beispiel: Führe die aktiven Ersetzungen in jedem dritten Satz ab Satz 1000 durch:

Befehl	Erklärung
S,10+10,/	
E,*A	Eintragen der Befehle in die Zeilen 10 bis 30
SUCH,*A+4	
TUE,10-30	
BREAK	Beendigung des Fremdstrings des Befehles S.
SUCH,1000	Positionieren auf Satz 1000
TUE,10-30□.	Durchführen der Ersetzungsvorschriften. Der Vorgang bricht ab, wenn der Befehl SUCH auf einen Fehler läuft, weil er am Ende der Datei oder am Ende des mittels BER eingestellten Dateibereiches angelangt ist. Man gibt dann bei diesem Fehler BEENDE ein.

e) Wie kann man aus mehreren Sätzen einen einzigen machen bzw. aus jeweils einem Satz mehrere?

Dieses Problem ist besonders interessant bei formatfreien FORTRAN-Quellen, bei denen mehrere Statements, die in getrennten Sätzen stehen, in einen Satz sollen oder umgekehrt jedes Statement in einen einzelnen Satz soll, wenn mehrere in jedem Satz stehen.

Auch solche relativ komplizierten Forderungen lassen sich mit AUFBEREITE erfüllen.

1. Beispiel: (für fortgeschrittene "AUFBEREITER")

Kette die Inhalte der Sätze 1000 bis 2000, durch Semikolon getrennt, hintereinander und schreibe den dadurch erzeugten String als einen Satz mit der Nummer 3000 in die Datei.

Befehlsfolge:

```
EIN,/
1␣ZENDE,255;S,3000,/
2␣Ⅱ255BREAKⅡ255ZENDE,21Ⅱ255=
BREAK
ZENDE,;
TUE,,1'1000-2000'2Ⅱ.
```

Als Voraussetzung gilt lediglich, daß als ZENDE das Zeichen "Newline" eingestellt war, und daß das Zeichen "DEL" mit Zentralcodewert 255 in den Sätzen nicht vorkommt - anderenfalls ersetze man 255 durch einen anderen Wert.

Der fortgeschrittene "Aufbereiter" möge sich selbst überlegen, warum dieses Beispiel so funktioniert.

Als Hinweis lediglich:

Der Befehl TUE bewirkt zunächst nur ein Verketten der gelesenen Sätze (mit dem aktuellen ZENDE als Trennzeichen), und erst wenn alle zu bearbeitenden Sätze im Eingabepuffer stehen, wird mit der Interpretation des ersten Befehles begonnen.

Zu beachten ist natürlich, daß die eingestellte Satzlänge (Befehl SATZLG) groß genug zur Aufnahme des erzeugten Satzes ist.

Zum Verständnis des Beispieles ist noch wichtig, daß der Befehl ZENDE noch mit dem vorher aktuellen Zeilenendezeichen abgeschlossen sein muß.

Nach Abarbeitung dieser Befehle ist wieder "Newline" mit Zentralcodewert 21 als ZENDE eingestellt, also der vorherige Zustand wiederhergestellt.

2. Beispiel: Zur Verdeutlichung der variablen Möglichkeiten von AUFBEREITE wollen wir auch zeigen, wie aus einem Satz mehrere Sätze gemacht werden können, also die Umkehrung des 1. Beispieles:

Trenne die Sätze 3000 bis 4000 bei jedem Semikolon auf und schreibe die so erzeugten Teilsätze ab Satz 10000 mit der Schrittweite 10 in die Datei (ohne das Semikolon hinter jedem Satz).

Befehlsfolge:

```
EIN,/
1␣S,10000+10,/
2␣BREAK;ZENDE,21
BREAK
ZENDE,;
TUE,,1'3000-4000'2⏎.
```

Soll das Semikolon hinter jedem Satz erhalten bleiben (z. B. bei einer formatfreien FORTRAN-Quelle, bei der jedes Statement in einen eigenen Satz der Datei soll), so genügt es, hinter der 4. Eingabezeile des Beispieles die zwei Eingabezeilen

```
DEF,1,H,/??;
ERS,1
```

einzufügen.

f) Mit Hilfe des Modus' WDH beim ERS-Befehl lassen sich auch kompliziertere Ersetzungsvorschriften durchführen.

Beispiel: Lösche in den Sätzen 100 bis 1000 alle Leerzeichen, die vor einem Semikolon stehen:

```
DEF,1,,/?␣;?;
ERS,1,WDH
E,100-1000⏎.
```

Die Ersetzung 1 ("Ersetze die Zeichenfolge <Leerzeichen><Semikolon> durch <Semikolon>) wird sooft wiederholt (MODUS=WDH), bis sie nicht mehr vorkommt.

g) Wie ersetzt man die Zeichenfolge "ANTON" überall durch "BERTA", außer wenn sie in Klammern steht?

Beispiel:

```
DEF,1,,/?(ANTON)?(ANTON)
DEF,2,,/?ANTON?BERTA
ERS,1'2
```

Solche "identischen Ersetzungen" wie die Ersetzung 1 sind natürlich nur sinnvoll, wenn nicht MODUS=WDH beim ERS-Befehl angegeben ist.

III STEUERUNGEN VON AUFBEREITE BEIM START

1. Das STARTE-Kommando

A) Mit ¤STARTE,AUFBEREITE,DEFINIERE

wird das Kommando AUFBEREITE definiert, das mit seinen Spezifikationen den komfortabelsten Zugang zu den Leistungen des Operators bietet.

B) Für Standardfälle hinsichtlich der zu bearbeitenden Dateien und verschiedenen Steuerungen empfiehlt sich die Form

¤STARTE,AUFBEREITE,DATEI=1- Datei¤.

da bei diesem Start einige Initialisierungszeit durch Startsatz-Aufbereitung etc. entfällt.

Unter der DATEN-Spezifikation des STARTE-Kommandos sind darüber hinaus bereits Befehle und Daten angebbar, die in jedem Fall interpretiert werden, bevor evtl. in den DIALOG-Betrieb umgeschaltet wird, so daß diese Form auch im BATCH-Betrieb benutzbar ist.

2. Das Kommando AUFBEREITE mit seinen Spezifikationen

Daneben gibt es das etwas bequemer benutztbare Kommando AUFBEREITE mit mehreren Spezifikationen, deren Beschreibung nun folgt.

Alle im Kommando angegebenen Werte sind dynamisch während des Laufes umstellbar, so daß z. B. beim Wechsel der Datei etc. der Programmlauf nicht beendet werden muß.

A.) DATEI

SPEZIFIKATIONSWERT	BEDEUTUNG
"undefiniert"	Es wird noch keine Datei angegeben, sondern der Dateiname wird mit dem DATEI-Befehl eingestellt.
(voreingestellt)	Im BATCH-Betrieb muß der DATEI-Befehl der erste der Information sein.
[<DBN>]. <DTN>[(GV-NR)]	Name der zu bearbeitenden Datei <DTN>, evtl. in Datenbasis <DBN>. Fehlt die Angabe der <GV-NR>, so wird die Datei mit der höchsten vorkommenden GV-NR. genommen. Eine externe Datei in der LFD oder auf WSP muß eingeschleust sein.

B.) MODUS

SPEZIFIKATIONSWERT	BEDEUTUNG
"undefiniert" (voreingestellt)	Die Einstellung des Arbeitsmodus' wird implizit aus den Kenndaten des Auftrages ermittelt, und zwar:

Gerät	Modus	
SIG100	SIVSAP	
SIG51/SIG50	SIG51	
5-kanal-FSR	FSR	
8-kanal-FSR mit Zeilenbreite = 240	VISTAR	(die Zeilenbreite wird dabei implizit auf 80 gesetzt)
sonst. 8-kanal FSR	FSR	
Wählgerät wie 8-kanal-FSR	FSR bzw. VISTAR	(durch Zeilenbreite im XTN-Kommando gesteuert)

SIVSAP	Es wird am Sichtgerät SIG100 gearbeitet mit Sichtgerätevermittler SIV100, also mit den Möglichkeiten des Schreibtafelmodus.
SIG100	Es wird am SIG100 ohne Sichtgerätevermittler gearbeitet, also rein fernschreiberanalog.
SIG51	Es wird am Sichtgerät SIG51 mit der Möglichkeit des Blockmodus gearbeitet.
FSR	Es wird fernschreiberanalog an irgendeinem Gerät gearbeitet.
VISTAR	Es wird am Infoton-Gerät VISTAR oder einem ähnlichen 8-kanal-Bildschirmgerät mit der Möglichkeit des Block-Modus gearbeitet.
SIG50	Es wird am SIG50 gearbeitet.

C) SATZLAENGE

SPEZIFIKATIONSWERT	BEDEUTUNG
"undefiniert" (voreingestellt)	Es werden Dateien mit maximal 161 Zeichen je Satz bearbeitet.
n natürliche Zahl	Es werden Dateien mit maximal n Zeichen je Satz bearbeitet. Ist $n < 161$, so wird n implizit auf 161 gesetzt.

Diese Angabe hat nichts mit der tatsächlichen Satzlänge der zu bearbeitenden Datei zutun, denn diese kann ja auch dynamisch gewechselt werden, sondern dient lediglich zur Kalkulation interner Puffergrößen, ist also mit für den Kernspeicherbedarf des Operators maßgeblich. Lediglich wenn die zu bearbeitende Datei eine genaue Satzlänge hat, die größer als n ist, so wird ein Fehler gemeldet.

Mit SATZLAENGE = n wird folgendes gesteuert:

a) Es können keine Sätze bearbeitet werden, deren Länge $>$ n ist.

b) Eingabesätze müssen $\geq$ n + 30 Zeichen sein.

c) Definierte Ersetzungen (siehe Befehl DEF) dürfen nicht länger als $\frac{n}{2}$ Zeichen sein.

Unabhängig davon besteht die Beschränkung, daß eine Konsoleingabe nicht länger als 6000 Zeichen sein darf, was aber nicht als wesentliche Behinderung erscheint.

D.) ZENDE

Angabe über das als Zeilenende-Markierung zu interpretierende Zeichen.

SPEZIFIKATIONSWERT	BEDEUTUNG
"undefiniert"	a) im Modus VISTAR: als Zeilenende wirkt das Zeichen " ← " mit dem Zentralcodewert 110, da im Blockmodus am VISTAR kein Zeilenwechsel mit übertragen wird. b) sonst: als Zeilenende wirkt das Zeichen "Newline" mit dem Zentralcodewert 21
(voreingestellt)	
/ Z	Als Zeilenende wirkt das im Normalstring nicht erlaubte Zeichen Z
Z	Als Zeilenende wirkt das Zeichen Z , das im Normalstring erlaubt ist.
<Ziffer>^{3}	Das Zeichen, dessen Zentralcodewert in Form von 3 Dezimalziffern angegeben ist, wird als Zeilenende interpretiert. Der Zentralcodewert muß zwischen 0 und 255 liegen.

Dieses Zeichen muß natürlich so gewählt werden, daß es nicht in den zu korrigierenden Sätzen vorkommt.

Die Möglichkeit, das Zeilenende-Kriterium frei einstellen zu können, hat 3 Vorteile:

a) Im Blockmodus am VISTAR wird bei der Eingabe kein Zeilenwechsel übertragen, so daß dort für einen Ersatz gesorgt werden mußte.

b) Grundsätzlich ist es vorteilhaft, das Zeilenendezeichen umsteuern zu können, wenn das normale "Newline"-Zeichen als Oktade in den Sätzen der zu bearbeitenden Datei vorkommt (Dateien vom Typ Ausgabezeichen etc.) oder in die Datei eingetragen werden soll.

c) Es gibt Geräte, die im Blockmodus andere Zeilen-Ende-Zeichen als "Newline" übergeben - z. B. ETX etc.

E.) PROTOKOLL (voreingestellt: A'KO)

SPEZIFIKATION	BEDEUTUNG
"undefiniert"	Keine globale Protokollsteuerung, d. h. die Aktionen des Operators werden nur protokolliert, wenn dem jeweiligen Befehl ein "P" vorangestellt wird.
A	Globale Protokollsteuerung, d. h. es wird immer protokolliert, außer wenn dem jeweiligen Befehl ein "N" vorangestellt wird.
KO	Alle Protokollierungen werden auf der Konsole und (bei eingeschaltetem Druckerprotokoll) auch im Ablaufprotokoll durchgeführt.
-STD-	Protokollierungen global voreingestellt für das Ablaufprotokoll.
E	Alle Eingabezeilen werden vor der Interpretation im Ablaufprotokoll protokolliert.
TIME	Vor jeder Anfrage nach Befehlen im Dialog wird die seit der letzten Anfrage (bzw. beim ersten Mal seit Operatorlaufbeginn) verbrauchte Rechenzeit ausgedruckt.
LDPROT	Alle Ladevorgänge werden protokolliert (zu Testzwecken).
V	Wenn ein Satz ersetzt, also überschrieben wird, wird auch sein vorheriger Inhalt protokolliert.

Mehrere Teilwerte sind durch Apostrophe zu trennen.

F.) MAL

SPEZIFIKATIONSWERT	BEDEUTUNG
"undefiniert"	Wird in der Konsoleingabe ein Fluchtsymbol erkannt, so wird es in das Zeichen FL mit dem Zentralcodewert 53 gewandelt.
/ Z	Als Mal wird das (im Normalstring nicht erlaubte) Zeichen Z interpretiert.
Z	Als Mal wird das Zeichen Z interpretiert.
<Ziffer>^{3}	Als Mal wird das Zeichen, dessen Zentralcodewert in Form von genau 3 Dezimalziffern angegeben ist, interpretiert.

Außer der Funktion, daß Fluchtsymbole der Konsoleingabe in das als Mal eingestellte Zeichen gewandelt werden, hat die Wahl des Mals den Effekt, daß bei eingeschalteter Umkodierung (siehe Befehl WANDEL) ein in der Datei gefundenes Mal bei der Ausgabe in das Zeichen FL umkodiert wird, also als das jeweilige codeabhängige Fluchtsymbol ausgegeben wird, während die übrigen gängigen Fluchtsymbole, da nicht im normalen Zeichensatz enthalten, in die Ersatzdarstellung umkodiert werden.

G.) INFORMATION

SPEZIFIKATIONSWERK	BEDEUTUNG
"undefiniert" (voreingestellt)	Alle Befehle werden im Dialog auf dem Terminal erfragt.
/<Fremdstring>	Zunächst werden alle Befehle und Daten des Fremdstrings interpretiert, bevor evtl. in den Dialogmodus umgeschaltet wird. Unabhängig vom eingestellten Modus werden die Befehle des Fremdstrings in Modus FSR mit dem Zeilenende-Zeichen "Newline" (ZC-Wert 21) interpretiert. Tritt innerhalb der Fremdstring-Abarbeitung ein Fehler auf, so kann dieser im Dialog vorrangig korrigiert werden, im Abschnitt führt er zum Abbruch des Operatorlaufes. Der Befehl K zum Korrigieren ist während der Fremdstringbearbeitung verboten, ebenso der Befehl FSRK.

H.) DEFINITIONEN

SPEZIFIKATIONSWERT	BEDEUTUNG
"undefiniert"	Es sind maximal 5 Ersetzungsvorschriften definierbar (siehe Befehl DEF), das heißt bei einem Ersetzungsvorgang können maximal 5 verschiedene Ersetzungen gleichzeitig
(voreingestellt)	durchgeführt werden.
n natürliche Zahl	Es sind maximal n verschiedene Ersetzungen gleichzeitig durchführbar. Ist $n \leq 0$ so wird es implizit auf 1 gesetzt.

Auch diese Angabe hat keinerlei Einfluß auf die tatsächliche Anzahl durchzuführender Ersetzungen, sondern ist lediglich eine obere Grenze für die Pufferkalkulation, die Einfluß auf den Kernspeicherbedarf hat.

Darüber hinaus hat einen weiteren Einfluß auf den Kernspeicherbedarf die tatsächliche Anzahl aktiver Ersetzungen (siehe Befehl ERS) sowie die maximale tatsächlich vorkommende Länge, deren obere Grenze wiederum durch SATZLAENGE/2 gegeben ist (siehe Spez. SATZLAENGE).

IV ALLGEMEINE BEMERKUNGEN

1. Es können Dateien vom Typ SEQ, RAN und RAM bearbeitet werden. Die maximal mögliche Satzlänge kann mit dem Befehl SATZLG oder der Kommando-Spezifikation SATZLAENGE eingestellt werden.

2. Es können nur Dateien bearbeitet werden, die Textfolgen enthalten. Das bedeutet:

   a) die Ganzworte der einzelnen Sätze müssen Typenkennung 3 haben;

   b) bei maximaler Satzlänge werden zu lange Sätze abgeschnitten,
   bei genauer Satzlänge werden zu kurze Sätze außerdem mit Blanks aufgefüllt.

3. Der Inhalt der Sätze wird nicht auf Übereinstimmung mit dem Dateityp verglichen, z. B. wird bei A-Dateien nicht überprüft, ob das erste Zeichen eines Satzes tatsächlich ein Vorschubsteuerzeichen ist.

4. Bei jeder Art von auftretenden E/A-Fehlern wird eine spezifizierte Fehlermeldung ausgedruckt (von S&SFR), der Operatorlauf jedoch nicht beendet.

5. Vorrangige Kommandos können nur eingegeben werden, wenn die Eingabeanforderung durch Befehl ¤: oder durch Datei= ¤: eingeleitet wird, denn in allen anderen Fällen werden auch Fluchtsymbole als normale Zeichen eingelesen.

6. Bei allen Befehlen, die sich auf bereits bestehende Datei-Inhalte beziehen (Duplizieren, Numerieren, Löschen, Korrigieren, Ersetzen, Protokollieren, Übertragen aus anderen Dateien) gelten dieselben Möglichkeiten der Zugriffsart:

   a) satzmarkenorientiert:

Satznummern	{dezimal / sedezimal}	volle 48 Bits
Satzmarken	in Oktadenform	
als Bereiche	<von> – <bis>	
als Relativangabe	<von> + <Anzahl>	

   b) assoziativ:

   Es wird über ein Suchkriterium zugegriffen. Dieses Suchkriterium kann sein:

   ein Suchstring

   ein Suchstring, auf einen bestimmten Spaltenbereich beschränkt.

   eine Spezifizierung der Satzlänge

   Durch den Befehl KRIT kann das Suchkriterium global erweitert werden.

   c) eine Mischform:

   Der assoziative Zugriff kann satzmarkenorientiert eingeschränkt werden. Eine Relativangabe " +<Anzahl>" bezieht sich dann auf die Anzahl laut Suchkriterium gefundener Sätze.

7. Es gibt 4 grundverschiedene Arten, Satzinhalte zu verändern:

a) Eine spezielle Möglichkeit, Blanks am Satzanfang und/oder Satzende oder ganz zu entfernen mit Hilfe einer globalen Einstellung durch den Befehl TRIM .

b) Eine Möglichkeit, assoziativ oder positionsgesteuert gezielt zu verändern mit Hilfe von aktiven definierten Ersetzungen.

Dazu gehören zum Beispiel:

α) eine bestimmte Zeichenfolge durch eine andere ersetzen,

β) diese Ersetzung auf einen Spaltenbereich zu beschränken,

γ) einen bestimmten Spaltenbereich durch einen angebbaren String zu ersetzen - unabhängig vom Inhalt,

δ) eine Zeichenfolge vor oder hinter Sätze ketten,

ε) eine Zeichenfolge vor oder hinter einer Spalte einfügen.

Dabei ist es möglich, in einem einzigen Durchlauf pro Satz beliebig viele Ersetzungen durchzuführen.

Definiert werden solche Ersetzungen mit dem Befehl DEF und aktiviert mit dem Befehl ERS.

Definierte, aktivierte Ersetzungen wirken bei allen dateiverändernden Befehlen - beim Eintragen ist AUFBEREITE somit als Eingabe-Makroprozessor benutzbar.

c) Durch definierbare Tabulator-Positionen können beliebige TAB-Funktionen durchgeführt werden.

d) Eine sehr wesentliche Korrekturmöglichkeit ist die Dialog-orientierte:

α) die Sichtfenster-Methode für das SIG100, das SIG51 sowie das VISTAR-Gerät mit den Möglichkeiten des Veränderns von Satzmarken und Satzinhalten, des Löschens, Einfügens und Duplizierens.

β) Die fernschreiberanaloge Korrektur:

Eine sehr bequeme Methode, an jedem beliebigen Terminal satzweise zu korrigieren mit den Funktionen:
Ersetzen und Löschen von Zeichenfolgen, Löschen des Satzes, Rückgängigmachen von Korrekturen, Interpretation von Backspace, Korrektur eines festlegbaren Ausschnittes von Sätzen etc.

8. Es besteht die Möglichkeit, mittels des Befehls WANDEL global zu steuern, ob Ausgaben umkodiert werden sollen, d. h. einzelne Zeichen umzukodieren bzw. nichtdruckbare Zeichen in dezimaler Ersatzdarstellung auszugeben (wichtig z. B. bei der Sichtfenstermethode des Korrigierens).

   Der eingestellte Code wird dabei implizit durch den Gerätetyp festgelegt, kann aber explizit mittels des Befehles CODTAB modifiziert werden.

9. Darüber hinaus kann die Ausgabe dahingehend manipuliert werden, daß der Benutzer explizit eine gewünschte Zeilenbreite seines Terminals einstellt (Befehl ZEILBR) festlegt, mit wievielen Ziffern Satznummern ausgegeben werden sollen, oder von jedem Satz nur eine maximale Zahl von Zeichen ausgeben läßt (z. B. nur die Satznummer) (Befehl KPROT).

10. Alle Satznummern können dezimal mit bis zu 14 Ziffern oder sedezimal durch ein H mit bis zu 12 folgenden Tetraden oder als (linksbündige) Oktadenfolge, in Ausrufezeichen eingeschlossen, eingegeben werden. Bei dezimalen oder sedezimalen Satznummern können führende Nullen weggelassen werden.

    Bei den Satznummern ist jedoch zu beachten, daß das mit dem Befehl BER eingestellte Arbeitsfenster der Datei (Teildatei) nicht überschritten wird.

11. Bei allen Operationen, die die Protokollierung von Sätzen zur Folge haben, gilt eine einheitliche Protokollregelung:

    a) Blanks am Zeilenende werden nicht mit ausgegeben.

    b) Das erste Zeichen der Zeile kennzeichnet, warum der Satz protokolliert wird - dabei bedeutet:

1. Zeichen	Grund der Protokollierung
"Blank"	Satz wurde eingetragen
P	einfache Protokollierung des vorhandenen Satzes
E	Satz ersetzt den vorher dort stehenden
K	Satz wurde korrigiert
L	Satz wurde gelöscht
D	Satz entstand durch Duplizierung
N	Satz entstand durch Umnumemerierung
V	Inhalt des Satzes vor einem überschreiben

    c) Das zweite Zeichen der Zeile ist ein Blank

d) Danach folgt die Satznummer, und zwar dezimal, wenn sie kleiner oder gleich 999999 ist, sonst sedezimal mit vorangestelltem H.

Bei dezimaler Ausgabe werden genausoviel Ziffern ausgegeben, wie mit dem Befehl SNRLNG eingestellt war - maximal 6.

Ist die Satznummer zu groß, um sie in der gewünschten Anzahl Ziffern darzustellen, so wird die Einstellung von SNRLNG implizit entsprechend erhöht.

Bei jedem Wechsel der Datei oder des Dateibereiches bzw. bei Programmbeginn wird SNRLNG implizit so eingestellt, daß die größte vorkommende Satznummer gerade darstellbar ist.

e) Hinter der Satznummer folgen ein Blank und der Satzinhalt, und zwar unter Maßgabe der eingestellten Zeilenbreite (Befehl ZEILBR) evtl. aufgebrochen. Teilzeilen werden dabei nicht eingerückt.

f) Die einzigen Abweichungen von dieser Protokollregelung sind:

α) Beim Korrigieren mit der Sichtfenstermethode werden Blanks am Zeilenende nicht unterdrückt.

β) Bei der fernschreiberanalogen Korrektur werden die ersten beiden Zeichen nicht vor der Satznummer eingefügt, ebenfalls bei der reinen Protokollierung mittels des Befehles P oder RP nicht.

γ) Bei der Protokollierung mit dem Befehl OP wird nur der Satzinhalt ausgegeben, bei A-Dateien wird dabei zusätzlich das erste Zeichen der Datei als Vorschubzeichen interpretiert. Enthält eine A-Datei Graphik- oder Text-Modus-Sätze für das SIG100 oder das SIG51, so werden diese entsprechend interpretiert.

12. Ein fehlender Spezifikationswert wird immer als " - ", also als "undefiniert" ergänzt (z. B. zwischen zwei aufeinanderfolgenden Kommata).

13. Die Keywordsteuerung ist nicht implementiert, man kann also nicht schreiben

<Spezifikationsname> = <Spezifikationswert> (z.B.: BEREICH = 1-100),

sondern die Spezifikationswerte werden nur über ihre Stellung den einzelnen Spezifikationen zugeordnet.

Die Spezifikationsnamen sind wichtig bei vorrangigen Korrekturen syntaktischer Fehler in Befehlen und beim Beschreibe-Befehl.

14. Grundsätzlich kann man AUFBEREITE zwischendurch anhalten, z. B. wenn man sich überlegt hat, daß die letzte Eingabe vielleicht doch nicht gut war, indem man mit ¤XAN¤. den Abwickler veranlaßt, mit¤¤ABW¤: eine Eingabe anzufordern, und indem man dann HALT,AUFBEREITE¤. eingibt.

    Vor dem nächsten Schreiben in die Datei, oder dem nächsten Lesen aus der Datei oder nach dem nächsten Protokollieren wird dies von AUFBEREITE bemerkt, und der Operator hört dann mit der laufenden Aktion auf, protokolliert die noch anstehende, nicht interpretierte Eingabe und fragt nach Befehlen in der Grundstufe.

15. Es existiert ein interner Zeiger auf einen "aktuellen Satz", der jeweils besetzt wird, wenn ein Satz aus irgendeinem Grund protokolliert wird - und zwar auf die Satznummer des protokollierten Satzes, der aber auch mittels des Befehles SUCH explizit gesetzt werden kann.

    Angesprochen wird der aktuelle Satz durch die Angabe *A statt einer Satznummer.

16. Ebenfalls eine Sonderstellung nimmt die Zeichenfolge *H als Satznummer ein. Sie kann überall dort, wo es sinnvoll ist, benutzt werden, um die höchste Satznummer der Datei +10 zu bezeichnen (z. B. beim Speichern, beim rückwärts Protokollieren etc.).

17. Außer in Fremdstring und anderen durch " / " eingeleiteten Strings sowie Satzinhalten werden Blanks grundsätzlich überlesen.

18. Beim Arbeiten mit AUFBEREITE im Abschnitt (von Lochkarten) durch Angabe der Befehle unter INFORMATION (Kdo. AUFBEREITE) oder DATEN (Kdo. STARTE,AUFBEREITE) ist darauf zu achten, daß Strings durch ¤) abgeschlossen werden müssen, wenn die restlichen Blanks einer Zeile nicht zum String gehören sollen (wichtig z. B. bei Ersetzungsstrings).

V INFORMALE BESCHREIBUNG DER SYNTAKTISCHEN ELEMENTE

Symbol	Wert	Bedeutung
⟨Bereich⟩	⟨Teilbereich⟩ ['⟨Teilbereich⟩]$^{\infty}$	beliebig viele ⟨Teilbereiche⟩, durch Apostroph getrennt, für den Satzmarken-orientierten Zugriff
⟨Teilbereich⟩	n	⟨Satznummer⟩ n
	n-	von ⟨Satznummer⟩ n an bis zum (Teil-)Dateiende
	-n	von (Teil-)Dateibeginn bis ⟨Satznummer⟩ n
	n-m	von ⟨Satznummer⟩ n bis ⟨Satznummer⟩ m
	n+m	von ⟨Satznummer⟩ n an m Sätze
⟨Satznummer⟩	⟨Ziffer⟩$_1^{14}$	dezimale Satznummer $< 2^{46}$
	H⟨Tetrade⟩$_1^{12}$	sedezimale Satznummer
	!⟨Oktade⟩$_1^6$!	Satzmarke in Oktadenform (Satznummern werden rechtsbündig interpretiert - führende Nullen können weggelassen werden). Lediglich die Oktadenform wird linksbündig interpretiert.
	*H	höchste existierende Satznummer der Datei + 10
	*A	aktuell eingestellte Satznummer (zeigt auf den zuletzt protokollierten oder mit "SUCH" eingestellten Satz).
⟨Numerierung⟩	n+m	von ⟨Satznummer⟩ n an mit ⟨Schrittweite⟩ m
	n	von ⟨Satznummer⟩ n an mit der zuletzt benutzten ⟨Schrittweite⟩ (voreingestellt: 10)
⟨Schrittweite⟩		dieselbe dezimale, sedezimale oder Oktaden-Form wie bei ⟨Satznummer⟩
⟨Suchkriterium⟩	"undefiniert"	Das Suchkriterium ist bei jedem Satz erfüllt
	⟨Spaltenbereich⟩	Das Suchkriterium ist erfüllt, wenn die letzte existierende Spalte des Satzes (Satzlänge) im ⟨Spaltenbereich⟩ liegt.
	⟨Suchstring⟩	Das Suchkriterium ist erfüllt, wenn der Suchstring im Satz enthalten ist.
	⟨Spaltenbereich⟩, ⟨Suchstring⟩	Das Suchkriterium ist erfüllt, wenn der Suchstring in dem angegebenen Spaltenbereich des Satzes enthalten ist.
	Zu beachten ist, daß das Suchkriterium aus den beiden letzten Spezifikationen des jeweiligen Befehles besteht.	
⟨Suchstring⟩	/⟨String⟩	⟨String⟩ ist eine beliebige Zeichenfolge, die durch das Zeilenende abgeschlossen wird und selbst kein Zeilenende enthält. Sie wird beim assoziativen Zugriff zum Suchen einer Zeichenfolge in einem (Teil-)Satz benutzt.

Symbol	Wert	Bedeutung
⟨Zugriff⟩	⟨Bereich⟩, ⟨Spaltenbereich⟩, ⟨Suchstring⟩	Dieses Symbol steht bei mehreren Befehlen für die drei Spezifikationen: BEREICH SPALTENBEREICH } (⟨Suchkriterium⟩) SUCHSTRING Mit Hilfe dieser 3 Spezifikationen wird bei fast allen Befehlen, die auf vorhandene Sätze zugreifen, die Zugriffsart spezifiziert, d. h. damit werden jeweils alle Sätze, die in einem der ⟨Teilbereiche⟩ von ⟨Bereich⟩ liegen, und für die das ⟨Suchkriterium⟩ erfüllt ist, angesprochen. (Zum⟨Suchkriterium⟩ siehe jedoch auch Befehl KRIT).
⟨Spaltenbereich⟩	n	⟨Spalte⟩ n
	n-	⟨Spalte⟩ n bis Satzende
	-n	Satzanfang bis ⟨Spalte⟩ n
	n-m	⟨Spalte⟩ n bis ⟨Spalte⟩ m
⟨Spalte⟩		dezimale Zahl n, 1 ≤ n ≤ maximale Satzlänge
⟨Ersetzung⟩	⟨E-Bereich⟩,/ ⟨E-String⟩	Es handelt sich um eine Ersetzungsvorschrift, die aus einem Ersetzungsbereich ⟨E-Bereich⟩ und einem Ersetzungsstring ⟨E-String⟩ besteht.
⟨E-Bereich⟩	⟨Spaltenbereich⟩	Der Ersetzungsbereich ist der angegebene ⟨Spaltenbereich⟩
	V	Der Ersetzungsbereich liegt vor dem Satz.
	V-n	Der Ersetzungsbereich liegt vor der angegebenen ⟨Spalte⟩n, also zwischen den ⟨Spalten⟩ n-1 und n.
	H	Der Ersetzungsbereich liegt hinter dem Satz.
	H-n	Der Ersetzungsbereich liegt hinter der ⟨Spalte⟩ n, also zwischen den ⟨Spalten⟩ n und n+1
	"undefiniert"	Der Ersetzungsbereich ist der gesamte Satz.
⟨E-String⟩	/⟨String⟩	Der Ersetzungstring bezeichnet, welche Ersetzung innerhalb des Ersetzungsbereiches durchgeführt werden soll. Durch das erste Zeichen des Ersetzungsstrings wird ein Trenner ⟨T⟩ festgelegt, der genau zweimal insgesamt im Ersetzungsstring vorkommen muß und beim zweiten Mal den Quellstring ⟨Q⟩ vom Zielstring ⟨Z⟩ trennt.
	Aufbau von ⟨String⟩ in diesem Fall:	
	⟨T⟩⟨Q⟩⟨T⟩⟨Z⟩	Im Ersetzungsbereich soll ⟨Q⟩ durch ⟨Z⟩ ersetzt werden.
	⟨T⟩⟨Q⟩⟨T⟩	Im Ersetzungsbereich soll ⟨Q⟩ gelöscht werden (Zielstring leer).
	⟨T⟩⟨T⟩⟨Z⟩	Der gesamte Ersetzungsbereich soll unabhängig vom Inhalt durch ⟨Z⟩ ersetzt werden (wenn Ersetzungsbereich = ⟨Spaltenbereich⟩), bzw. ⟨Z⟩ soll an der durch den Ersetzungsbereich spezifizierten Stelle eingefügt werden (wenn Ersetzungsbereich = V, H, V-n oder H-n). (Quellstring leer).

Symbol	Wert	Bedeutung
	<T><T>	Der gesamte Ersetzungsbereich soll unabhängig vom Inhalt gelöscht werden (nur erlaubt, wenn der Ersetzungsbereich = <Spaltenbereich>). (Quellstring und Zielstring leer).
<z>	<Z2> oder <T><Z2>	beliebige Zeichenfolge, in der <T> nicht vorkommt. In diesem Fall muß <Z2> aus einer geraden Anzahl von Tetraden bestehen, die paarweise zu Oktaden gewandelt werden.
<Spezialzeichen>	Z	Das <Spezialzeichen> ist das Zeichen Z .
	<Ziffer>^{3}	Das <Spezialzeichen> ist das Zeichen mit dem in 3 Dezimalziffern angegebenen Zentralcodewert.
<Fremdstring>	/<S>	Ein Fremdstring besteht aus beliebig vielen Zeilen, wird also nicht durch Zeilenende beendet. Leerzeilen werden überlesen. Ein <Fremdstring> wird beendet durch: 1.) Eingabeende oder 2.) eine Zeile, die nur aus der Zeichenfolge BREAK besteht.
"undefiniert"		Sonder-Spezifikationswert, der entweder durch ein Minuszeichen oder durch das Weglassen eines Spezifikationswertes angegeben wird. Weggelassen wird ein Spezifikationswert durch die Aufeinanderfolge zweier Kommata oder durch Fortlassen von Spezifikationswerten am Zeilenende.

VI BESCHREIBUNG ALLER BEFEHLE

A) Dynamische Veränderung der im Kommando AUFBEREITE festgelegten Werte

1.) DATEI

Wechseln der aktuell bearbeiteten Datei

1 Spezifikation:

DATEI=[<dbn>.]<dtn>[(GV-Nr.)]

Wirkungen:

a) Die aktuelle bearbeitete Datei ist von nun an die Datei <dtn>, evtl. in der Datenbasis <dbn>; wenn nicht durch die Generationsversionsnummer spezifiziert, die jenige mit der höchsten GV-Nr.

b) Wenn das globale Protokoll eingeschaltet ist, so werden die wichtigsten Daten der Datei aufgelistet, und zwar in der Form:
<Kat.>.<dtn>(GV-Nr.),<Typ>,<Satzbau>,<Satzzahl>,<erster Satz>-<letzter Satz>

Handelt es sich um eine LF-Datei oder eine WSP-Datei, die nicht im Standard-DMK liegt, so wird für <Kat> das LFD-BKZ bzw. das DMK eingesetzt, anderenfalls die Datenbasis.

c) Alle definierten Ersetzungen werden inaktiv, bleiben aber definiert (siehe Befehl ERS).

d) Es wird der Standard-Dateibereich eingestellt: 1-999999 (siehe Befehl BER).

e) Die Protkoll-Länge für Satznummern wird so eingestellt, daß die höchste existierende Satznummer noch dargestellt werden kann (siehe Befehl SNRLNG).

f) Die "Systemsicherungen" werden ausgeschaltet (siehe Befehle SYSI/AEND).

g) Es wird implizit "TRIM,-" eingestellt (siehe Befehl TRIM)

2.) PROT

Einstellen des globalen Protokolls

1 Spezifikation: PROTOKOLL
Teilwerte zugelassen

Erlaubte Spezifikationswerte identisch wie bei der Kommando-Spezifikation PROTOKOLL. Auflistung der erlaubten Werte durch "INF,PROT".

Wirkung: siehe Kdo-Spezifikation PROTOKOLL

3.) MAL

Einstellen des als Fluchtsymbol zu interpretierenden Zeichens

1 Spezifikation: MAL

zugelassene Werte: <Spezialzeichen>

Wirkung: siehe Kommando-Spezifikation MAL.

4.) MODUS

Einstellen des gewünschten Arbeitsmodus'

1 Spezifikation: MODUS

zugelassene Werte: SIVSAP
SIG100
SIG51
SIG50
VISTAR
FSR

Wirkung: Es wird die Art der Fehlerbehandlung sowie die Behandlung des Korrekturmodus' für das spezifizierte Gerät gesteuert (siehe auch Kommando-Spezifikation MODUS).

5.) SATZLG

Umstellen der maximal bearbeitbaren Satzlänge

1 Spezifikation: SATZLAENGE

mögliche Werte: natürliche Zahl n

$n \geq 161$

Wirkung: siehe Kommando-Spezifikation SATZLAENGE

Darüber hinaus ist zu beachten, daß bei diesem Befehl durch die dafür notwendige Arbeitsspeicher-Reorganisation alle definierten Ersetzungen verlorengehen

6.) MAXDEF

Umstellen der maximal möglichen Anzahl von definierten Ersetzungen

1 Spezifikation: DEFINITIONEN

mögliche Werte: natürliche Zahl n

$n \geq 1$

Wirkung: siehe Kommando-Spezifikation DEFINITONEN

Auch hierbei gilt die Einschränkung wie beim Befehl SATZLG, daß alle bisher definierten Ersetzungen verlorengehen.

7.) ZENDE

Einstellen des als Zeilenende zu interpretierenden Zeichens

1 Spezifikation: ZENDE

zulässiger Wert: <Spezialzeichen>

Wirkung: siehe Kommando-Spezifikation ZENDE

B) INFORMIEREDIENSTE

1.) INF

Allgemeiner Informieredienst

1 Spezifikation: MODUS (Teilwerte erlaubt)

MODUS	Bedeutung
INF	Auflisten aller für MODUS zugelassenen Werte beim Befehl INF
SIT	Auflisten der aktuellen Situation, d.h. aller aktuell eingestellten Steuergrößen etc.
DATEI	Informieren über die wichtigen Daten der aktuell bearbeiteten Datei
BEF	Auflisten aller erlaubten Befehle
PROT	Informieren über eingestelltes Protokoll
SYSI	Auflisten aller beim Befehl SYSI zugelassenen MODI
ERS	Auflisten aller definierten (evtl. aktiven) Ersetzungen
TAB	Informieren über definierte Tabulatoren
n	Informieren über internen Namen n

2.) B

Dieser Befehl dient zum Beschreiben von Befehlen bzw. einzelnen Spezifikationswerten bestimmter Befehle.

1. Spezifikation: NAME (Teilwerte zulässig)

Zulässige Werte	Bedeutung
BEF	Beschreibung aller Befehle von AUFBEREITE
<Befehlsname>	Beschreibung des angegebenen Befehls
<Spezifikationsname> (<Befehlsname>)	Beschreibung des angegebenen Spezifikationswertes des Befehls
BEISPIELE(<Befehlsname>)	Ausdrucken von Beispielen für den angegebenen Befehl

Alle Werte, also Befehlsname und Spezifikationsname, dürfen im Rahmen der Eindeutigkeit abgekürzt werden.

2. Spezifikation: MODUS (Teilwerte zulässig)

Zulässige Werte	Bedeutung
"undefiniert"	Ausdrucken einer Kurzbeschreibung des Befehls
KURZ	wie "undefiniert"
NKO	Beschreibung nicht auf Konsole, sondern nur ins Ablaufprotokoll. Davor und dahinter wird ein Seitenvorschub generiert.
PAGE	Vor jedem Teilwert von NAME wird ein Seitenvorschub generiert. Bei BEF vor jedem neuen Befehl.
SPEZ	Ausdrucken einer Liste der erlaubten Spezifikationsnamen für den angegebenen Befehl.
A	Ausdrucken einer kompletten Beschreibung der angegebenen Befehle einschließlich Spezifikationsliste, Beispielen, Kurzbeschreibung etc.

C) PROTOKOLLIERDIENSTE

1.) P

Protokollieren von Sätzen

1., 2. und 3. Spezifikation: <Zugriff>

Wirkung: Alle Sätze unter <Bereich>, für die die Suchbedingung erfüllt ist, werden unabhängig vom aktuell eingestellten Protokollzustand auf dem Terminal und, falls das Ablaufprotokoll eingeschaltet ist, auch in diesem protokolliert, und zwar in der Form:

<Satznummer><Blank><Satzinhalt>

Die Länge der <Satznummer> ist dabei abhängig von der mit dem Befehl SNRLNG eingestellten Größe, falls die Satznummer $\leq$ 999999 ist, sonst wird sie dargestellt als H<Tetrade>^{12}.

2.) OP

Ohne Satznummer Protokollieren von Sätzen

1.2. und 3. Spezifikation: <Zugriff>

Wirkung: Alle Sätze, die laut Suchbedingung in <Bereich> gefunden werden, werden auf dem Terminal protokolliert, und zwar ohne Satznummer nach folgendem Prinzip:

a) Es handelt sich um eine O-Datei:

Die Sätze werden ganz normal wie mit dem Befehl P protokolliert - nur ohne Vorschaltung der Satznummern.

b) Es handelt sich um eine A-Datei:

Das Vorschubsteuerzeichen wird interpretiert:

α) Der Satz ist ein Schreibtafel- oder Graphiksatz für das SIG50/SIG51/SIG100: er wird als solcher nur auf dem Terminal ausgegeben.

β) Ansonsten wird der komplette Satz - ohne Kürzung oder Aufbrechen unter Interpretation des 1. Zeichens als Vorschubsteuerzeichen auf das Terminal ausgegeben.

Bei A-Dateien wird nicht umkodiert - der Satz wird unverändert ausgegeben. Bei MODUS=FSR (siehe Befehl MODUS) wird auch ins Ablaufprotokoll ausgegeben, falls dieses eingeschaltet ist.

3.) RP

Rückwärts Protokollieren von Sätzen

1. Spezifikation: SATZNUMMER

erlaubte Spezifikationswerte	Bedeutung
<Satznummer> + <Anzahl>	Von <Satznummer> an werden rückwärts, d. h. zum Dateianfang hin, <Anzahl> Sätze protokolliert. Der Satz <Satznummer> muß dabei nicht existieren.
* H + <Anzahl>	Die letzten <Anzahl> Sätze der Datei werden rückwärts protokolliert.

2. und 3. Spezifikation: <Suchkriterium>

Wirkung: Von der angegebenen Satznummer an werden rückwärts (zum Dateianfang hin) <Anzahl> Sätze protokolliert, die die Suchbedingung erfüllen.

Typische Anwendungsfälle: "Welches ist der letzte Satz vor dem Satz mit der Nummer 100000?"
oder: "Wo tritt zum letzten Mal in der Datei die Zeichenfolge ANTON auf?".

Fehlt <Anzahl>, so wird es zu 1 ergänzt.

D) DATEIVERÄNDERNDE BEFEHLE

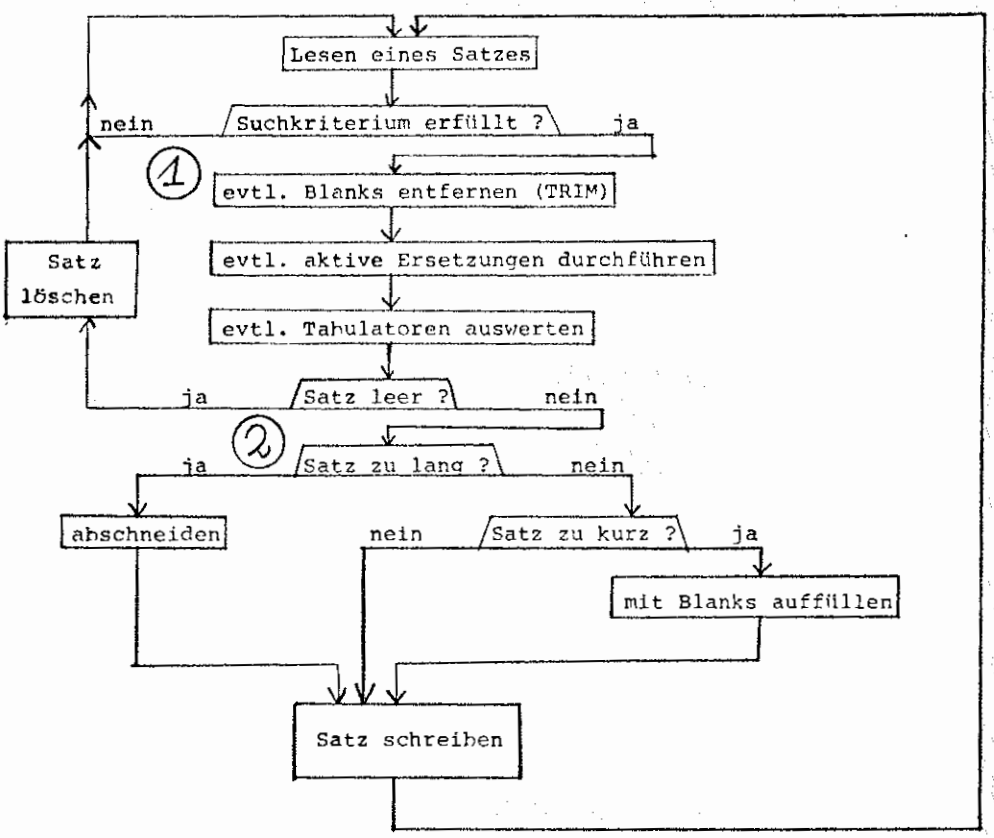
Alle dateiverändernden Befehle haben folgende Merkmale gemeinsam:

a) Welche Sätze bearbeitet werden sollen, steuert der Benutzer über den <Zugriff>, also über die Spezifikationen

BEREICH
SPALTENBEREICH
SUCHSTRING .

b) Durch ein dem eigentlichen Befehlsnamen vorangestelltes "N" bzw. "P" kann die globale Protokolleinstellung umgangen werden.

c) Das Vorgehen ist vereinfacht so:



Unterbrochen wird dieser Kreis, wenn beim Lesen eines Satzes der jeweilige unter <Zugriff> spezifizierte Bereich verlassen wird. Beim Löschen wird bei (1) natürlich bereits gelöscht.

Beim Korrigieren wird bei (2) das Vorlegen zur Korrektur und das Einlesen der Korrekturen zwischengeschaltet.

1.) L / NL / PL

Löschen von Sätzen

1., 2. und 3. Spezifikation : <Zugriff>

Die Sätze, die in <Bereich> laut Suchbedingung gefunden werden, werden - evtl. mit Protokoll - gelöscht. Wenn mit Protokollierung gelöscht werden soll, so wird jeder Satz einzeln gelöscht und anschließend protokolliert, sonst werden, falls kein <Suchkriterium> spezifiziert wurde, die <Teilbereich>e, die nicht durch eine Relativangabe (m + n) gekennzeichnet sind, als Ganzes gelöscht (<von> - <bis>).

2.) EIN / NEIN / PEIN

Eintragen von Sätzen

1 Spezifikation: INFORMATION

erlaubte Spezifikationswerte: { "undefiniert" / <Fremdstring> }

Wirkung:

a) INFORMATION = "undefiniert"

Die Information wird mit "TEXT␣:" am Terminal erfragt (nur im Dialog erlaubt).

Eingegebene Fluchtsymbole werden dabei als Zeichen gelesen und leiten keine Programmiersystem Kommandos ein (siehe Befehl MAL).

b) INFORMATION=<Fremdstring>

Die Zeilen des <Fremdstrings> sind bereits die Information.

In beiden Fällen wird die Information durch Eingabeende oder eine Zeile, die nur aus der Zeichenfolge

BREAK

besteht, beendet.

Der Befehl EIN dient zum Eintragen von Sätzen mit vorangestellter <Satznummer>.

Die INFORMATION wird zeilenweise abgearbeitet - jede Zeile muß nach folgenden Konventionen aufgebaut sein:

a) Eintragen einzelner Sätze:

<Satznummer><Blank><Satzinhalt>

Die <Satznummer> kann dabei in beliebiger Form eingegeben werden - durch das Trennzeichen <Blank> ist die Eindeutigkeit gegeben.

b) Duplizieren eines einzelnen Satzes:

<Satznummer> D

Durch diese Zeile wird der Inhalt des zuletzt darüberstehenden Satzes in den mit der angegebenen <Satznummer> dupliziert.

Diese Möglichkeit erlaubt es also, bequem einen bestimmten Satzinhalt in mehrere verschiedene Sätze einzutragen, ohne ihn mehrmals hinschreiben zu müssen.

Beispiel:

```
EIN/
100␣ANTON
200D
300D␤.
```

Die Zeichenfolge ANTON wird in die Sätze 100, 200 und 300 eingetragen.

c) Löschen eines einzelnen Satz- Teilbereiches :

<Satznummer>L (Der angegebene Satz wird gelöscht).

oder

<Satznummer>-<Satznummer>L (Der angegebene Satzbereich wird gelöscht).

Aktive Ersetzungen wirken bei diesem Befehl auf die Eingabezeilen (siehe Befehl ERS).

3.) S / NS / PS

Speichern von Sätzen in die Datei

1. Spezifikation: NUMERIERUNG
erlaubte Spezifikationswerte:<Numerierung>

2. Spezifikation: INFORMATION
erlaubte Spezifikationswerte { <Fremdstring> / "undefiniert" }

Wirkung:

a) INFORMATION=<Fremdstring>

Die Zeilen des <Fremdstring>'s werden sequentiell gemäß NUMERIERUNG eingetragen.

Nach <Fremdstring>-Ende wird wieder in den Befehls-Modus umgeschaltet, und es werden neue Befehle gelesen.

b) INFORMATION= "undefiniert"
(Nur im Dialog erlaubt)

Evtl. noch vorhandene weitere Eingabezeilen werden ignoriert, und der einzutragende Text wird mit TEXT␤: auf dem Terminal angefordert.

Dabei wird in einen anderen Eingabemodus umgeschaltet, der zwei Konsequenzen hat:

α) Fluchtsymbole im Text werden als solche gelesen, in das aktuelle Mal (siehe Befehl MAL) umgewandelt und mit dem Text eingetragen - werden also nicht als vorrangige Kommando-Einleitung interpretiert.

β) Bei Eingabeende wird nicht in den normalen Befehlsmodus zurückgeschaltet, sondern erneut Text angefordert.
Beendet wird dieser Eingabemodus nur durch eine leere Eingabe oder durch eine Zeile, die nur aus der Zeichenfolge BREAK besteht.

Wird bei einer erneuten Textanforderung weiterer Text eingegeben, so läuft die angegebene <Numerierung> normal weiter.

Leere Zeilen bei der Eingabe werden überlesen. In den Modi FSR und SIG100 wird das Errozeichen (siehe Befehl ERRZEI) interpretiert. Aktive Veränderungen (siehe Befehl TRIM, TAB und ERS) werden zwischen Eingabe und dem Schreiben in die Datei nach Art eines Eingabe-Makro-Prozessors ausgeführt.

4.) FSRK

Fernschreiber-Korrektur-Modus

1.2. und 3. Spezifikation: <Zugriff>

Dieser Befehl dient zum Korrigieren

a) an Terminals, die nicht über einen Blockmodus verfügen und/oder

b) von Dateien, die im Blockmodus schlecht oder gar nicht korrigierbar sind (sehr lange Sätze; Zeichen mit dem Zentralcodewert 33 in den Sätzen enthalten etc.)

Außerdem gibt es Korrektur-Fälle, bei denen dieser Modus trotz der größeren Zahl der erforderlichen Konsolzyklen (mindestens einer pro Satz) bequemer und schneller ist als das Blockmodus-Korrigieren (z. B. Einfügen einer bestimmten ziemlich langen Zeichenfolge in mehreren Sätzen etc.)

Das wesentliche an diesem Korrekturmodus ist, daß jeder Satz einzeln zur Korrektur vorgelegt wird, und zwar in der Form:

<Satznummer>␣<Satzinhalt>

␣:

Dabei wird die Eingabeforderung immer so gestellt, daß das erste eingegebene Zeichen unter das Blank zwischen Satznummer und Satzinhalt zu stehen kommt, denn das erste eingegebene Zeichen hat eine Sonderfunktion und gehört nicht zur Information als Satzinhalt.

Eine besondere Rolle spielen Leerzeilen:

1 Leerzeile in der Eingabe beendet genau wie eine leere Eingabe die Korrektur des aktuellen Satzes und schaltet evtl. zur Korrektur des nächsten unter <Zugriff> spezifizierten Satzes.

2 Leerzeilen beenden den gesamten Korrekturvorgang und schalten in den normalen Befehlseingabemodus zurück - der Rest der Eingabe wird überlesen.

Grundsätzlich ist zu beachten, daß eine Eingabe nur als Korrekturvorschrift zu einem einzigen Satz ausgewertet wird.

Tritt in einer Eingabe eine Zeile auf, die nur aus der Zeichenfolge

BREAK

besteht, so wird diese Zeile interpretiert wie 2 Leerzeilen mit anschließendem Zurückschalten in den Befehlseingabemodus zuzüglich einer simulierten Anfrage nach Befehlen. Der Korrekturmodus wird also beendet (noch anstehende Korrekturen des aktuellen Satzes werden noch durchgeführt), und es können zur Einsparung eines Konsolzyklus' sofort Befehle angeschlossen werden.

Diese Möglichkeiten erlauben es, zusammen mit den eigentlichen Korrektureingaben, die im folgenden beschrieben werden, in jedem Fall mit einem Konsolzyklus pro Korrektur eines Satzes auszukommen - auch wenn mehrere Sätze nacheinander korrigiert werden sollen.

Die Korrektureingaben:

Die wesentlichste Zeile der Korrektureingabe ist zunächst die erste eingegebene Zeile. In ihr zählt jedes Zeichen genau für das darüberstehende Originalzeichen der zur Korrektur vorgelegten Zeile.

Dabei sind folgende Fälle zu unterscheiden für ein eingegebenes Korrekturzeichen (außer dem ersten Zeichen der ersten Zeile):

a) Es handelt sich um das IDENTITÄTSZEICHEN:
   das darüberstehende Zeichen bleibt unverändert erhalten.

Das IDENTITÄTSZEICHEN wird in jeder Korrektureingabe neu definiert durch das erste Zeichen der ersten Korrekturzeile, das unter dem Blank zwischen Satznummer und Satzinhalt steht.

b) Es handelt sich um das LOESCHZEICHEN:
das darüberstehende Zeichen wird aus dem Satz entfernt.

Ist das LOESCHZEICHEN gleichzeitig das letzte Zeichen der ersten Korrekturzeile, so wird der gesamte Rest des vorgelegten Satzes gelöscht.

Das LOESCHZEICHEN wird global durch den Befehl LOEZEI definiert (siehe dort).

c) Es handelt sich um das ERSETZUNGSZEICHEN:
das darüberstehende Zeichen wird durch die nächste komplette Eingabezeile, die noch nicht interpretiert wurde, ersetzt. Sind schon alle Eingabezeilen abgearbeitet, so wird die letzte interpretierte für alle noch anstehenden ERSETZUNGS-ZEICHEN herangezogen.

Das ERSETZUNGSZEICHEN wird global durch den Befehl ERSZEI definiert (siehe dort).

d) In allen anderen Fällen ersetzt das Zeichen der ersten Korrektureingabezeile das darüberstehende Zeichen.

Zur Sicherung der spaltengerechten Eingabe in der ersten Korrektureingabezeile wird in den Modi FSR, VISTAR und SIG100 natürlich auch hier das ERRORZEICHEN interpretiert (siehe Befehl ERRZEI); dies ist sogar der wesentlichste Anwendungsfall für das ERRORZEICHEN!

Das erste Zeichen der ersten Korrektureingabezeile hat folgende Sonderfunktionen:

a) Es handelt sich um das LOESCHZEICHEN:
alle bisher am vorgelegten Satz durchgeführten Änderungen - sei es durch definierte, aktive Ersetzungen, die vor der Vorlage des Satzes wirken (siehe Befehl ERS), sei es durch vorhergehende Korrekturen, die nicht durch mindestens eine Leerzeile abgeschlossen wurden (in diesem Fall wird ja der korrigierte Satz noch einmal vorgelegt) - werden rückgängig gemacht.

Wird dieses LOESCHZEICHEN direkt von einer Leerzeile gefolgt, so wird die Korrektur des Satzes sofort abgebrochen, und er wird in keinem Fall zurückgeschrieben; anderenfalls wird der Satz in Originalform (evtl. nach erneuter Durchführung der aktiven Ersetzungen) wieder zur Korrektur vorgelegt.

b) Es handelt sich um das ERSETZUNGSZEICHEN:
Damit können direkt Ersetzungen spezifiziert werden, die überall im Satz durchgeführt werden - in diesem Fall wird nur diese erste Eingabezeile interpretiert.

Sei zum Beispiel "<" das definierte Ersetzungszeichen:

```
100␣ANTON␣&␣BERTA
␣□:<␣<**□.
100␣ANTON***&***BERTA
␣□:<A<□.
100␣NTON***&***BERT
```

Die zwischen dem ersten und dem zweiten Ersetzungszeichen stehende Zeichenfolge wird also durch die hinter dem zweiten stehende ersetzt (evtl. gelöscht).

5.) K / NK / PK

Gerätespezifisches Korrigieren

1., 2. und 3. Spezifikation: <Zugriff>

Bei diesem Befehl wird zunächst anhand des implizit (über den Gerätetyp des Terminals) oder explizit (Befehl MODUS) eingestellten Modus' verzweigt:

Handelt es sich nicht um den Modus SIG50, SIG51, SIG100 oder VISTAR, so wird sofort zum Befehl FSRK verzweigt, also zur fernschreiberanalogen Korrektur, da dann davon ausgegangen wird, daß das Terminal nicht über einen Blockmodus verfügt.

Im anderen Falle werden die unter <Zugriff> spezifizierten Sätze auf den Bildschirm geschrieben, und es wird eine Eingabe angefordert.
Der Benutzer hat nun die Möglichkeit, mittels Bildschirmmanipulationen wie DELETE/INSERT ect. den Bildschirminhalt zu verändern.

Zulässige Veränderungen sind dabei:

a) Löschen von Sätzen

b) Hinzufügen neuer Sätze

c) Verändern von Satzinhalten

d) Verändern von Satznummern

Danach kann der veränderte Bildschirminhalt wieder abgeschickt werden.

Für diese Eingabe gelten dieselben Konventionen wie beim Befehl EIN, also

<Satznummer><Blank><Satzinhalt>

pro Eingabezeile.

Auch die Sonderzeilen zum Löschen und Duplizieren (weglassen des Blanks hinter der Satznummer) können genauso wie beim Befehl EIN eingefügt werden.

Der Operator AUFBEREITE vergleicht dann die eingegebenen Zeilen mit den (intern gespeicherten) ausgegeben und führt alle Änderungen, die auf dem Bildschirm mit Hardwarefunktionen gemacht wurden, auch in der Datei durch.

Das heißt:

a) Sätze, die nicht wieder eingelesen werden, werden auch in der Datei gelöscht.

b) Hinzugekommene Sätze werden in die Datei eingetragen.

c) Veränderte Sätze werden in die Datei zurückgeschrieben.

Zusätzlich zu den Sonderfunktionen, die beim Befehl EIN möglich sind, gibt es noch 2 spezielle Formen von Eingabezeilen:

a) Eine Zeile, die nur aus den zwei Zeichen NL besteht, bewirkt, daß nicht wiedereingelesene Sätze nicht in der Datei gelöscht werden (in der gesamten Eingabe).

b) Eine Zeile, die nur aus

<Satznummer> N

besteht (ohne Blank hinter <Satznummer>), bewirkt, daß der Satz mit der entsprechenden Satznummer nicht verändert oder gelöscht wird. Das ist nützlich, wenn Änderungen des Satzes auf dem Bildschirm durchgeführt wurden, die dann doch nicht übernommen werden sollen.

Wird eine Zeile eingelesen, die nicht den Konventionen genügt, also zum Beispiel eine unzulässige Satznummer enthält oder kein trennendes Blank zwischen Satznummer und Satzinhalt, so wird diese Zeile protokolliert und eine "vorrangige Korrektur-Eingabe" angefordert.

Diese vorrangige Korrektur-Eingabe kann wiederum aus mehreren Zeilen bestehen, die den Eingabe-Konventionen des K-Befehles genügen. Der fehlerhaft eingegebene Satz kann also z. B. korrekt wieder eingegeben werden. Das hat den Vorteil, daß solche durch Bildschirm-Manipulationen evtl. "verdorbenen" Sätze nicht unbedingt gelöscht werden, weil sie natürlich in der Eingabe nicht wiedergefunden werden konnten.

Das liegt daran, daß nach der Interpretation aller Eingabezeilen erst überprüft wird, welche Sätze nicht wieder eingelesen wurden.

Die vorrangige Korrektur-Eingabe wird dabei ganz schlicht als ein kompletter String an die Stelle eingefügt, an der die unzulässige Eingabezeile stand.

6.) D / ND / PD

Duplizieren von Sätzen

1. Spezifikation: NUMERIERUNG
erlaubte Spezifikationswerte: <Numerierung>

2., 3. und 4. Spezifikation: <Zugriff>

Wirkung:

Jeder Satz, der laut Suchbedingung gefunden wird, wird nach dem Einlesen zunächst evtl. verändert (siehe Befehle TRIM und ERS) und dann im Kernspeicher zwischengespeichert.

Erst wenn auf diese Weise alle zu duplizierenden Sätze eingelesen wurden, wird mit der eigentlichen Duplizierung gemäß der angegebenen Numerierung begonnen.

Dieses Vorgehen hat zwei wesentliche Vorteile gegenüber dem sofortigen Duplizieren:

a) Wenn ohne Protokollierung (Befehl ND bzw. Befehl D bei ausgeschaltetem Globalprotokoll) dupliziert wird, ist das Verfahren sehr schnell, da viele Positionierungen und Betriebswechsel entfallen.

b) Quell- und Zielbereich dürfen sich beliebig überschneiden.

7.) NUM / NNUM / PNUM

Umnumerieren von Sätzen

1. Spezifikation: NUMERIERUNG
erlaubte Spezifikationswerte: <Numerierung>

1., 3. und 4. Spezifikation: <Zugriff>

Wirkung:

Die Vorgehensweise ist dieselbe wie beim Duplizieren (siehe Befehl D): zunächst alle Sätze aller Quellbereiche in den Kernspeicher lesen, dann gemäß NUMERIERUNG duplizieren und anschließend die Quellsätze löschen, falls sie nicht schon bei der Duplizierung überschrieben wurden dadurch, daß sich Quell- und Zielbereiche überschnitten.

Es ist jedoch zusätzlich zu beachten, daß unter folgenden Bedingungen eine Überschneidung von Quell- und Zielbereichen verboten ist, da die Quellsätze nicht exakt gelöscht werden können:

1. Durch die Duplizierung entstehen Sätze mit Satznummern $\geq 2^{36}$ (=68719476736) und
2. die Schrittweite der Numerierung ist $\neq 1$

Sind diese beiden Bedingungen erfüllt, so können aus internen Arithmetik-Gründen keine Quellsätze der Umnumerierung gelöscht werden.

8.) E / NE / PE

Ersetzungen durchführen

1., 2. und 3. Spezifikation: <Zugriff>

Wirkung:

Die durch <Zugriff> spezifizierten Sätze werden

a) gelesen
b) verändert (siehe Befehle TRIM, TAB und ERS)
c) zurückgeschrieben und
d) evtl. protokolliert

Erhält ein Satz durch die Veränderung die Länge 0, so wird er explizit gelöscht.

Ein Satz, der zwar laut Suchbedingungen gefunden, aber nicht verändert wurde (wegen nicht zutreffender <Ersetzung>)wird nicht zurückgeschrieben, sondern nur protokolliert, und auch das nur, wenn beim ERS-Befehl nicht MODUS=NP angegeben war (siehe Befehl ERS).

9.) GET / NGET / PGET

Dieser Befehl dient zum Übertragen von Sätzen aus einer beliebigen Datei in die aktuell bearbeitete Datei - evtl. unter Veränderung durch aktive Ersetzungen etc.

1. Spezifikation: DATEI
   [<dbn>.]<dtn>
2. Spezifikation: <Numerierung>
3., 4. und 5. Spezifikation: <Zugriff>

Die Übertragung geschieht satzweise mit oder ohne Protokollierung.

Ist DATEI= "undefiniert", so wird aus der aktuell bearbeiteten Datei in diese selbst übertragen (gemäß <Numerierung>), also wie beim Befehl D/ND/PD , jedoch ohne Kernspeicherzwischenpufferung. Dabei dürfen sich Quell- und Zielbereiche nicht überschneiden.

Ist NUMERIERUNG= "undefiniert", so wird bei der Übertragung die Numerierung der Quelldatei übernommen - DATEI und NUMERIERUNG sollten also nicht beide "undefiniert" sein.

E) Globale Einstellung bzw. Aktivierung von Änderungsvorschriften für die dateiverändernden Befehle

1.) TRIM

Steuerung zu entfernender Blanks

1. Spezifikation: MODUS
Teilwerte zulässig

Zulässige Werte	Wirkung
"undefiniert"	Es werden keine Blanks entfernt.
V	Blanks am Satzanfang werden entfernt.
H	Blanks am Satzende werden entfernt.
A	alle Blanks im Satz werden entfernt.

Mit diesem Befehl wird eine globale Steuerung eingestellt, die im weiteren bei allen dateiverändernden Befehlen wirkt, und zwar genauso wie aktive definierte Ersetzungen. Diese Steuerung wirkt jedoch in jedem Fall vor Ausführung aktiver Ersetzungen.

Beim Einstellen auf eine andere Datei wird die globale Einstellung immer auf "undefiniert" gesetzt.

2.) DEF

Definieren von Ersetzungen

1. Spezifikation: INAM

erlaubte Spezifikationswerte: natürliche Zahl n, $1 \leq n \leq$ maximale Ersetzungszahl (siehe Befehl MAXDEF)

2. und 3. Spezifikation: <Ersetzung>

Wirkung:

Unter dem "internen Namen" n wird eine Ersetzungsvorschrift gespeichert, die später beliebig aktiv oder inaktiv gemacht werden kann (siehe Befehl ERS).

Zu Beginn ist eine solche Ersetzungsvorschrift inaktiv, es sei denn, es wird ein interner Name neu definiert, der vorher schon existierte und bereits aktiv war.

Ist eine Ersetzung syntaktisch unzulässig, so wird sie komplett neu angefordert zur Korrektur - es müssen also beide Spezifikationen neu eingegeben werden. Der Ersetzungsstring darf maximal halb so lang sein wie die maximale Satzlänge (siehe Befehl SATZLG).

Es gibt folgende Möglichkeiten für Ersetzungsvorschriften:

a) Ersetzen einer bestimmten Zeichenfolge, falls vorhanden, durch eine andere. Das kann darüber hinaus noch eingeschränkt werden durch eine Spaltenangabe: Ersetzung nur dort, wo der zu ersetzende String komplett in dem angegebenen ⟨Spaltenbereich⟩ liegt.

Beispiel: DEF,1,20-40,/?MAX?ANTON
Ersetzung von "MAX" durch "ANTON" und zwar nur im ⟨Spaltenbereich⟩ 20-40.

"?" ist in diesem Fall der Trenner, und die gesamte Ersetzungsvorschrift soll auf dem "internen Namen" 1 gespeichert werden.

b) Löschen einer bestimmten Zeichenfolge, falls vorhanden:

DEF,2,-15,/?BERTA?
Die Zeichenfolge "BERTA" soll überall in den ersten 15 Spalten gelöscht werden, da der einzusetzende String leer ist.

c) Einfügen einer Zeichenfolge an einer bestimmten Stelle des Satzes. Dazu muß der Quellstring des Ersetzungsstrings leer sein (Ersetzungsstring muß als erste 2 Zeichen die beiden Trenner enthalten).

Beispiel: DEF,3,V-5,/??␣␣␣
Vor Spalte 5 sollen 3 Blanks eingefügt werden.

d) Ersetzen eines bestimmten Spaltenbereiches des Satzes durch eine Zeichenfolge - unabhängig vom Inhalt des Spaltenbereiches.

Beispiel: DEF,4,7-10,/??␣
Der gesamte Spaltenbereich 7-10 soll durch ein einziges Blank ersetzt werden.

Bei dieser Art der Ersetzung wird jeder Satz, der so kurz ist, daß er die erste angegebene Spalte nicht mehr enthält, zunächst durch Auffüllen mit Blanks auf die erforderliche Minimallänge gebracht - im Beispiel bis Spalte 6 einschließlich mindestens verlängert, falls er kürzer war.

Damit lassen sich zum Beispiel Sätze auf eine bestimmte genaue Länge bringen:

DEF,5,73-,/??

Diese Ersetzungsvorschrift bewirkt:

Alle Sätze, die kürzer als 72 Zeichen sind, werden zunächst bis Spalte 72 mit Blanks aufgefüllt. Alles, was hinter Spalte 72 folgt, wird dann ersetzt - in diesem Fall durch den Leerstring, also gelöscht.

Für eine genaue Beschreibung der Ersetzung siehe V.

Zu beachten ist, daß durch die bloße Definition mit dem Befehl DEF zunächst nur eine Speicherung der Ersetzungsvorschrift erfolgt, noch keine Veränderung von Sätzen.

Erst wenn eine Ersetzung mit dem Befehl ERS aktiviert wird, ist sie bei jedem dateiverändernden Befehl wirksam (siehe Befehl ERS).

Beginnt der einzusetzende String wiederum mit dem Trennzeichen, so wird die Zeichenfolge dahinter als Tetradenfolge interpretiert, um beliebige Zeichen erzeugen zu können.

Beispiel: DEF,1,,/?&??AF15AF
Diese Definition bewirkt die Ersetzung von "&" durch die Zeichenfolge: Blank-Newline-Blank.

3.) ERS / NERS / PERS

Ersetzungen aktivieren

1. Spezifikation: AKTIV
Teilwerte erlaubt

erlaubte Spezifikationswerte	Wirkung
"undefiniert"	Alle definierten Ersetzungen werden inaktiv - die restlichen Spezifikationen werden ignoriert.
n (natürliche Zahl)	Die unter dem "internen Namen" n definierte Ersetzung wird aktiviert. Zusätzliche Bedingung: Anzahl der Teilwerte ≤ maximale Ersetzungszahl (siehe Befehl MAXDEF)
-STD-	Alle definierten Ersetzungen werden der Reihe nach aktiviert.

2. Spezifikation: MODUS
Teilwerte erlaubt

erlaubte Spezifikationswerte	Wirkung
"undefiniert"	Ersetzungen werden nur in den Satzteilen durchgeführt, in denen noch keine Ersetzungen vorher durchgeführt wurden (Normalfall). Alle Sätze, die beim Befehl E (siehe dort) laut Suchbedingung gefunden werden, werden protokolliert, auch wenn sie nicht verändert wurden.
NP	Nur veränderte Sätze werden beim Befehl E protokolliert.
WDH	Ersetzungen werden rekursiv durchgeführt, d. h. jede Ersetzung wird in dem gesamten evtl. durch eine vorhergehende Ersetzung erzeugten Satz durchgeführt (rechenzeitaufwendig).

3. Spezifikation: ANZAHL

erlaubte Spezifikationswerte	Wirkung
"undefiniert"	Jede Ersetzung wird so oft durchgeführt, bis die zu ersetzende Zeichenfolge nicht mehr im Satz vorkommt.
$n \geq 1$	Jede Ersetzung wird maximal n-mal pro Satz durchgeführt (vorne im Satz beginnend).

Allgemeines:

Mit Hilfe des Befehls ERS werden definierte Ersetzungen (siehe Befehl DEF) aktiviert.

So aktivierte Ersetzungen wirken bei den Befehlen:

E,K,FSRK,S,EIN,GET,D und NUM.

Die Ersetzungen werden dabei in der Reihenfolge durchgeführt, in der sie unter AKTIV angegeben werden - alle dort nicht angegebenen Ersetzungen werden inaktiv.

Insbesondere kann ein "interner Name" mehrmals unter AKTIV aufgeführt sein - die zugehörige Ersetzung wird dann auch mehrmals durchgeführt. Insgesamt dürfen

jedoch nicht mehr Teilwerte unter AKTIV angegeben werden - also nicht mehr aktive Ersetzungen vorhanden sein - als durch die maximale Ersetzungsanzahl erlaubt ist (siehe Befehl MAXDEF).

Mit der Spezifikation ANZAHL kann die Zahl der Durchführungen pro Ersetzung und pro Satz beschränkt werden - in diesem Fall ist es auch sinnvoll, eine Ersetzung mehrmals unter AKTIV aufzuführen, wenn z.B. zwei Ersetzungen jeweils höchstens einmal durchgeführt werden sollen, eine andere jedoch dreimal:

```
ERS,1'3'2'2'2,,1    (Die Ersetzungen 1 und 3 werden
                     jeweils höchstens 1-mal, die Ersetzung 2
                     höchstens 3-mal pro Satz durchgeführt).
```

Um den Modus WDH zu verstehen, muß man die Vorgehensweise kennen:

Wenn festgestellt wird, daß an einer bestimmten Stelle eine Zeichenfolge <Q> durch eine andere Zeichenfolge <Z> ersetzt werden soll, wird zunächst die zu ersetzende Zeichenfolge <Q> gelöscht und dann durch einen Verweis auf <Z> ersetzt. Erst wenn alle Ersetzungen für den Satz durchgeführt worden sind, werden diese Verweise interpretiert und der neue Satz aus ihnen und dem übriggebliebenen Teil von <Q> zusammengesetzt.

Dadurch wird verhindert, daß eine Zeichenfolge, die durch eine Ersetzung entstanden ist, durch eine weitere Ersetzung noch einmal verändert wird.

Ein Beispiel: Der Satz 10: "Eine KatzE" sei gegeben.

Durch die Befehle:
```
DEF,1,,/?E?e
DEF,2,,/?Eine?DIE
ERS,2'1
E,10
```

wird der Satz 10 so verändert: "DIE Katze".

Will man jedoch genau diesen Effekt ausnutzen, daß eine Ersetzung auch auf bereits durch Ersetzungen entstandene Zeichenfolgen wirkt, dann kann man das durch MODUS=WDH erreichen.

Ein Beispiel: Man möchte alle Blanks, die vor einem Semikolon stehen, auf maximal eines reduzieren.

Das erreicht man durch:
```
DEF,1,,/?␣␣;?␣;
ERS,1,WDH
```

Dadurch, daß bei der rekursiven Ersetzung jedesmal nach einem "Durchgang" durch den Satz der neu entstandene Satz generiert werden muß und nicht mit Verweistechniken gearbeitet werden kann, ist dieses Verfahren natürlich recht zeitaufwendig.

Verboten ist es, Definitionen zu aktivieren, die unabhängig vom Inhalt der Spaltenbereiche, Ersetzungen, Einfügungen oder Löschungen durchführen, und deren Spaltenbereiche sich berühren oder gar überschneiden.

Ebenso dürfen solche Definitionen nicht hinter anderen aktiviert werden, die evtl. in deren Spaltenbereichen Änderungen vorgenommen haben. Solche Vorgehensweisen werden jedoch erst als fehlerhaft erkannt und führen dann zum Abbruch des aktuellen Vorganges, wenn durch sie Unstimmigkeiten in der Verweisstruktur auftreten - im Modus WDH gehen sie im allgemeinen gut.

Bei rekursiven Ersetzungen ist besondere Sorgfalt darauf zu verwenden, keine unendlichen Schleifen zu produzieren.

Es ist darauf zu achten, daß auch zwischenzeitlich kein Satz länger wird als die maximale Satzlänge gestattet (siehe Befehl SATZLG).

Nun zur Art und Weise, wie die Ersetzungen, die aktiv sind, bei den verschiedenen Befehlen wirken:

a) Bei den Befehlen E, D, GET und NUM:
   Die Sätze werden aus der Datei eingelesen, und wenn die Suchbedingung erfüllt ist, werden zunächst evtl. Blanks entfernt (siehe Befehl TRIM). Dann werden die Ersetzungen durchgeführt. Danach werden Tabulatoren ausgewertet und anschließend die Sätze zurückgeschrieben (bei E nur die veränderten, bei D und NUM alle, aber mit veränderter Numerierung).

b) Bei den Befehlen EIN und S:
   Hier ist die Vorgehensweise ähnlich, nur daß die Sätze nicht aus der Datei, sondern aus dem ⟨Fremdstring⟩ oder direkt aus einer speziellen Konsoleingabe eingelesen werden. Ersetzungen wirken also dabei nach Art eines Eingabe-Makro-Prozessors.

c) Bei den Befehlen K und FSRK:
   Anders ist es hierbei: die Sätze werden zunächst aus der Datei gelesen, auf die Suchbedingung überprüft und gemäß TRIM-Einstellung und aktiven Ersetzungen verändert. Erst danach werden sie zur Korrektur auf dem Terminal vorgelegt - also bereits in veränderter Form.

   Beim Wiedereinlesen spielen die aktiven Ersetzungen und die TRIM-Einstellung keine Rolle mehr - es wird nur festgestellt, daß die wieder eingelesenen Sätze sich von den Originalsätzen unterscheiden. Insbesondere wirken also bei dem Befehl K aktive Ersetzungen nicht auf neu hinzugekommene Sätze.

Stellt man fest, daß die definierten Ersetzungen eigentlich für gewisse Sätze beim Befehl K oder FSRK nicht durchgeführt werden sollten, so kann man das folgendermaßen verhindern:

a) bei der Korrektur im Sichtfenstermodus:
Man schreibe direkt hinter die entsprechende Satznummer (also ohne trennendes Blank) ein N (für nicht korrigieren) und lösche den Rest des Satzes auf dem Bildschirm.

b) bei der satzweisen Fernschreiber-Korrektur (siehe Befehl FSRK):
Man gebe als einziges Zeichen das LOESCHZEICHEN ein und sofort dahinter einen oder mehrere Zeilenvorschübe zum Abschluß der Korrektur des jeweiligen Satzes.

4.) TAB

Mit Hilfe dieses Befehles können Tabulatoren und ein Tabulatorzeichen eingestellt werden.

1. Spezifikation: POSITIONEN:
max. 20 Teilwerte zulässig: n, $1 \leq n \leq$ max. Satzlänge
Die angegebenen natürlichen Zahlen setzen alle Tabulatorpositionen neu fest. Ist POSITIONEN "undefiniert", so werden alle Tabulatoren gelöscht.

2. Spezifikation: TABZEICHEN
erlaubter Wert: /<Fremdstring>
Der <Fremdstring> muß genau ein Zeichen enthalten, das bei seinem Auftreten ein Vorrücken zur nächsten Tabulatorposition bewirkt. Ist die aktuelle Spaltenposition schon größer als die höchste Tabulatorposoition, so wird ein Blank eingesetzt.

Die eingestellten Tabulatoren wirken bei allen dateiverändernden Befehlen, und zwar nach evtl. aktiven Ersetzungen.

F) GLOBALE EINSTELLUNGEN VON SONDERZEICHEN UND SONDERFUNKTIONEN

1.) LOEZEI

Löschzeichen einstellen

1 Spezifikation: LOESCHZEICHEN
erlaubte Spezifikationswerte: <Spezialzeichen>

Wirkung:

Dieses Zeichen ist nur für den Fernschreiber-Korrekturmodus (siehe Befehle K und FSRK) von Bedeutung.

Diejenigen Zeichen des zur Korrektur vorgelegten Satzes, unter die das LOESCHZEICHEN geschrieben wird, werden gelöscht.

Ist das Löschzeichen das letzte Zeichen der ersten Korrektur-Eingabezeile, so wird der gesamte Rest des Satzes gelöscht.

Ist das Löschzeichen das erste Zeichen einer Korrektureingabe für einen Satz, der zum zweitenmal zur Korrektur vorgelegt wurde (weil noch keine Leerzeile in der vorigen Korrektur aufgetreten war), so werden alle bisherigen Änderungen des Satzes rückgängig gemacht und der Satz in der Originalform wieder zur Korrektur vorgelegt. Wird hierbei in derselben Eingabe die Korrektur des Satzes durch Leerzeilen abgeschlossen, so wird der Satz auf keinen Fall korrigiert - auch nicht, wenn er vor der Vorlage schon durch aktive Ersetzungen verändert worden war (siehe Befehl ERS).

Voreingestellt ist zu Beginn:

beim MODUS	das Zeichen
8-Kanal-FSR	} mit dem ZC-Wert 168
5-Kanal-FSR	] mit dem ZC-Wert 163
sonst	> mit dem ZC-Wert 156

2.) ERSZEI

Ersetzungszeichen einstellen

1 Spezifikation: ERSETZUNGSZEICHEN
erlaubte Spezifikationswerte: <Spezialzeichen>

Wirkung:

Das Ersetzungszeichen ist nur im Fernschreiber-Korrekturmodus (siehe Befehle K und FSRK) von Bedeutung.

Jedes Zeichen des zur Korrektur vorgelegten Satzes, unter das in der ersten Korrektureingabezeile ein Ersetzungszeichen geschrieben wird, wird durch die komplette nächste Eingabezeile ersetzt, die noch nicht interpretiert wurde - also pro Ersetzungszeichen in der Reihenfolge des Auftretens eine Eingabezeile. Sind mehr Ersetzungszeichen als Zeilen in der Eingabe, so wird für alle überschüssigen Ersetzungszeichen jeweils die letzte Eingabezeile genommen.

Voreingestellt ist zu Beginn:

im MODUS	das Zeichen
8-Kanal-FSR	{ mit dem ZC-Wert 191
5-Kanal-FSR	[mit dem ZC-Wert 162
sonst	< mit dem ZC-Wert 155

Ist das Ersetzungszeichen das erste Zeichen der ersten Eingabezeile, so können hiermit direkt Ersetzungen angegeben werden in der Form:

<Ersetzungszeichen><zu ersetzender String><Ersetzungszeichen><einzusetzender String>

3.) ERRZEI

Error-Zeichen einstellen

1 Spezifikation: ERRORZEICHEN
erlaubte Spezifikationswerte: < Spezialzeichen >

Wirkung:

Das Errorzeichen ist nur für die Modi FSR, VISTAR und SIG100 relevant. In diesen Modi wird es zum Überschreiben ungültiger Zeichen in der Backspace-Funktion interpretiert.

Zweckmäßigerweise wird man, wenn vorhanden, das Backspace als Errorzeichen definieren (es ist auch mit dem Zentralcodewert 44 voreingestellt - beim Modus VISTAR ist jedoch 32 voreingestellt).

Da es Geräte mit anderen BS-Ersatzzeichen gibt, ist das Errorzeichen frei wählbar. Diese Error-Funktion ist besonders wichtig im Fernschreiber-Korrekturmodus (siehe Befehle K und FSRK), um trotz Fehlerkorrektur in der Eingabe spaltengerecht weiterarbeiten zu können, aber auch sonst ist es bequemer als das ␣'. Man kann mit dieser Funktion natürlich nicht über den Zeilenanfang hinausgehen - weitere Error-Zeichen werden ignoriert.

4.) ZEILBR

Einstellen der gewünschten Zeilenbreite des Terminals

1 Spezifikation: ZEILENBREITE

zulässiger Wert: natürliche Zahl n
$1 \leq n \leq$ maximale Satzlänge

Mit diesem Befehl kann die implizit durch die Geräteart des Terminals, also vom System her vorgebene Zeilenbreite explizit verändert werden. Sollen Zeilen protokolliert werden, die länger sind als die aktuell eingestellte Zeilenbreite, so werden sie aufgebrochen und in Teilzeilen ausgegeben, jedoch ohne Einrücken der Folgezeilen.

(Eine eingestellte Protokoll-Kürzung wirkt aber vor dem Aufbrechen aufgrund der Zeilenbreite - siehe Befehl KPROT).

5.) KPROT

Kürzung protokollierter Sätze

1 Spezifikation: MAXLAENGE

erlaubte Spezifikationswerte	Wirkung
"undefiniert"	Alle protokollierten Sätze werden einschließlich Satznummer in ihrer tatsächlichen Länge, jedoch ohne Blanks am Zeilenende protokolliert.
natürliche Zahl $n > 0$	Soll ein Satz protokolliert werden, so wird er zunächst abgeschnitten, falls er einschließlich Satznummer länger als n Zeichen ist. Erst danach werden Blanks am Zeilenende entfernt und dann wird evtl. noch der Satz aufgebrochen, falls er nach dem Abschneiden noch länger als die aktuelle Zeilenbreite ist (siehe Befehl ZEILBR). Damit läßt sich z. B. erreichen, daß nur die Satznummern protokolliert werden.

6.) BER

Defintion eines Bereiches der aktuellen Datei als Teildatei

1 Spezifikation: BEREICH
voreingestellt: 1-999999

mögliche Werte	Wirkung
"undefiniert"	der Bereich ist die gesamte Datei, also alle Satznummern von 1 bis $2^{48} - 1$
-⟨Satznummer⟩	der Bereich geht vom Dateianfang bis ⟨Satznummer⟩
⟨Satznummer⟩-	der Bereich geht von ⟨Satznummer⟩ bis Dateiende
n - m	der Bereich geht von ⟨Satznummer⟩ n bis ⟨Satznummer⟩ m

Zu Beginn des Programms und bei jedem DATEI-Befehl wird der aktuelle Teildatei-Bereich wieder auf die Voreinstellung 1 - 999999 gesetzt (Standardwerte für Texthaltungsdateien).

Die Einstellung einer Teildatei hat 2 Wirkungen:

a) außerhalb des eingestellten Bereiches wird jeder Zugriff auf die Datei als fehlerhaft zurückgewiesen:
Dadurch kann man einen Schutz vor versehentlicher Zerstörung von Infomation erreichen.

b) Jede ⟨Satznummer⟩ wird sowohl bei der Eingabe (z. B. in Befehlen) als auch bei der Protokollierung mit der ersten Satznummer translatiert. Dadurch läßt sich unter Umständen das unbequeme Arbeiten mit großen Satznummern vermeiden. Fehlt die erste Satznummer oder ist sie = 1, so wird nicht translatiert.

Besonders vorteilhaft ist das z. B. beim Arbeiten mit Dateien, in denen Quellen mittels RB&QUELLHALT [4] gehalten werden, da man sehr bequem eine solche Quelle, die verändert werden soll, als Teildatei auswählen kann.

Die Befehle BER und KONV sind die einzigen, bei denen eine eingestellte Translation oder Begrenzung nicht wirken.

7.) KORBER

Korrekturbereich von Sätzen definieren

1 Spezifikation: KORREKTURBEREICH

erlaubte Spezifikationswerte	Wirkung
"undefiniert"	Die Sätze werden komplett zur Korrektur vorgelegt.
m-	Die Sätze werden von Spalte m an bis zum Satzende zur Korrektur vorgelegt.
-n	Die Sätze werden vom Anfang bis Spalte n vorgelegt.
m-n	Die Sätze werden nur von Spalte m bis Spalte n zur Korrektur vorgelegt.

Die Einstellung eines Korrekturbereiches ist nur für den Fernschreiber-Korrekturmodus (siehe Befehl FSRK) wirksam.

Normalerweise werden dabei die Sätze immer komplett zur Korrektur vorgelegt. Will man jedoch in vielen Sätzen immer nur im selben kleinen Spaltenbereich korrigieren oder will man sehr lange Sätze (z. B. > 1000 Zeichen) korrigieren, empfiehlt es sich, aus Zeitersparungsgründen nur den zu korrigierenden Satzausschnitt vorlegen zu lassen und zu korrigieren. Alles was links und rechts von diesem Ausschnitt im Satz ist, bleibt unverändert - auch wenn der Ausschnitt selbst durch die Korrektur länger oder kürzer wird. Bei mehrfach aufeinanderfolgender Vorlage desselben Satzes wird der Satzausschnitt jedesmal neu bestimmt.

8.) KRIT

Globale Beeinflussung von Suchkriterien

1. Spezifikation: MODUS

erlaubte Spezifikationswerte	Wirkung
-STD- (voreingestellt)	Eine Suchbedingung ist erfüllt, wenn das <Suchkriterium> erfüllt ist, d. h. wenn z. B. der <Suchstring> im Satz enthalten ist.
"undefiniert"	Hierdurch wird ein <Suchkriterium> negiert, eine Suchbedingung ist also dann erfüllt, wenn das <Suchkriterium> nicht erfüllt ist.

Mit Hilfe dieses Befehls ist es möglich, als Suchbedingung z. B. zu spezifizieren:

Protokolliere alle Zeilen, in denen in den ersten 8 Spalten kein "E" vorkommt:

```
KRIT,-
P,1-,-8,/E
```

2. Spezifikation: GLOBALKRITERIUM

Erlaubter Spezifikationswert ist entweder "undefiniert" (dann wird ein früher spezifiziertes Globalkriterium gelöscht) oder ein Fremdstring bis zum Zeilenende, der eine Maske beschreibt, die zusätzlich zu einer im jeweiligen Befehl angegebenen Suchbedingung auf die zu bearbeitenden Sätze anwendbar sein muß, um das Suchkriterium zu erfüllen.

Die "Maske" besteht aus mehreren Zeichenfolgen, die durch spezielle Zeichen mit Sonderfunktionen getrennt sind.

Es gibt zwei solche Spezialzeichen:

das Blankzeichen und
das Beliebigzeichen.

Dort, wo das Blankzeichen in der Maske auftaucht, darf im zu untersuchenden Satz eine beliebige lange (auch leere) Folge von Blanks stehen, und wo das Beliebigzeichen auftritt, eine beliebige (auch leere) Zeichenfolge. Die Teile der "Maske", die jeweils durch eines der beiden Sonderzeichen getrennt werden, müssen in der richtigen Reihenfolge unverändert im zu untersuchenden Satz vorhanden sein.

Die beiden Sonderzeichen werden durch die beiden ersten Zeichen des als GLOBALKRITERIUM angegebenen Fremdstrings definiert, und zwar definiert das erste Zeichen das Blankzeichen und das zweite Zeichen das Beliebigzeichen.

Beispiele: Der leichten Lesbarkeit wegen wird in den folgenden Beispielen immer das Blank als "Blankzeichen" und der Punkt als "Beliebzeichen" genommen.

Aufbau des Fremdstrings unter GLOBALKRITERIUM	Erläuterung des dadurch entstehenden Suchkriteriums
/␣.ANTON.BERTA	Es wird zunächst die Zeichenfolge "ANTON" gesucht, danach eine beliebige, beliebig lange (auch leere) Zeichenfolge, und danach die Zeichenfolge "BERTA"
/␣.FELD(.)	Es wird die Benutzung von Feldelementen des FORTRAN-Felds mit Namen "FELD" gesucht, also der Name "FELD", gefolgt von einer öffnenden Indexkammer, danach eine beliebige Zeichenfolge als Index und danach die schließende Indexklammer.
/␣.M␣O␣R␣I␣T␣Z	Es wird der Name "MORITZ" gesucht, unabhängig davon, wieviele Blanks darin eingestreut sind. Deshalb wird nach jedem Zeichen eine beliebige Folge von Leerzeichen zugelassen.
/␣.:=␣ARR[.]␣;	Es werden Sätze einer ALGOL60-Quelle gesucht, in denen die Werte von Elementen des Arrays ARR einer Variablen zugewiesen werden. Das heißt, es wird überprüft, ob zunächst das Zuweisungszeichen ":=" enthalten ist, danach evtl. eine Folge von Blanks, danach die Zeichenfolge "ARR[", dann eine beliebige Zeichenfolge als Indexausdruck, darauf die schließende Indexklammer und anschließend (evtl. nach einer Folge von Blanks) das Semikolon als Statement-Begrenzer.

Komplette Beispiele:

α) Es werden FORTRAN-Statements gesucht, die eine arithmetische IF-Anweisung enthalten und keine Anweisungsnummer in Spalte 1 bis 5:

```
KRIT,-STD-,/␣.I␣F␣(.).,.,
P,1-,-5,/␣␣␣␣␣
```

Gefunden würden dabei Sätze der Form

```
IF  (A B) 100 , 200 , 300
```

aber auch

```
IF(BOLL(I(J),K,L))RETURN
```

aber z. B. nicht mehr:

```
IF(B(I,J,K))RETURN
```

oder

```
IFAX(I)=(I,J,K)
```

oder

```
100   IF(A)1,2,3
```

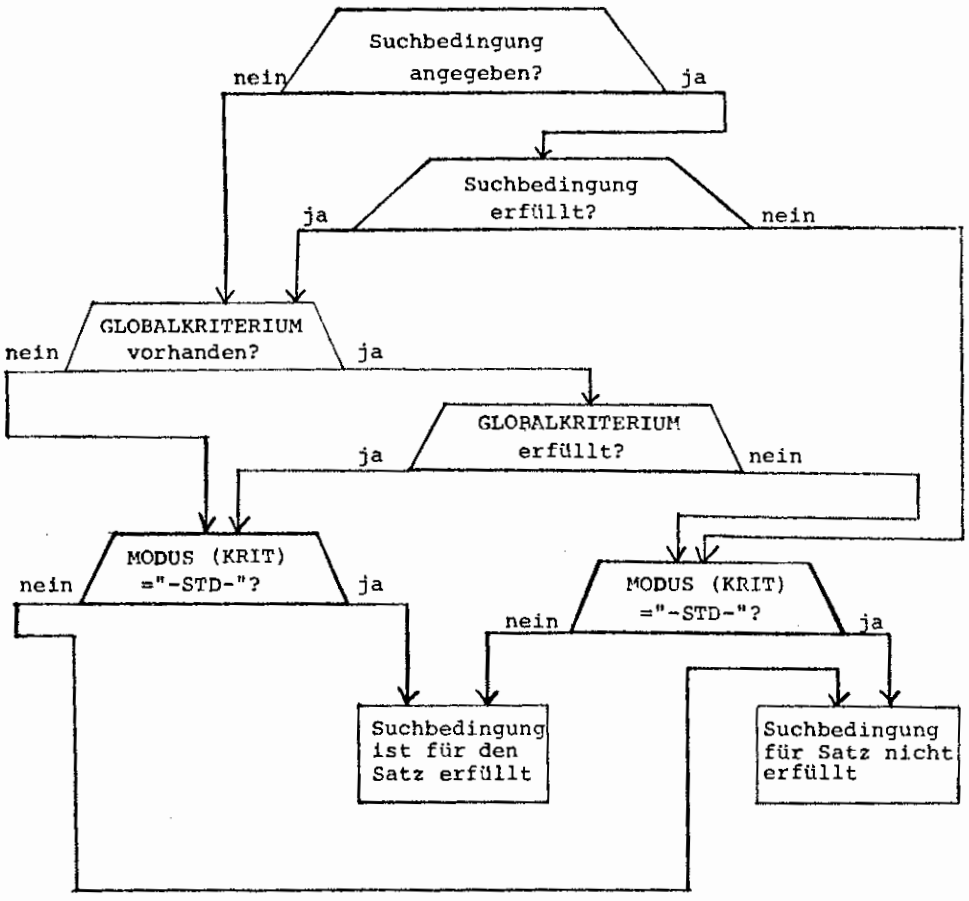
β) Es sollen alle for-Statements einer ALGOL60-Quelle korrigiert werden:

```
KRIT,-STD-,/␣.'␣F␣O␣R␣'.:=
K,1-
```

γ) Es soll die Deklaration der ALGOL60-Prozedur FUNK mit 2 Parametern gesucht werden.

```
KRIT,-STD-,/␣.'␣P␣R␣O␣C␣E␣D␣U␣R␣E␣'␣F␣U␣N␣K␣(.,.)␣;
P,1+1
```

Das Vorgehen beim Auswerten von Suchkriterien ist folgendes:



Einzige Ausnahme: Wenn weder eine Suchbedingung bei dem Befehl noch ein GLOBALKRITERIUM angegeben sind, dann wird der eingestellte MODUS von KRIT ignoriert, da sonst keine Sätze gefunden würden - in diesem Fall sollen aber wohl alle Sätze die Suchbedingung erfüllen.

9.) SNRLNG

LÄnge der Satznummern einstellen

1 Spezifikation: SNRLAENGE

erlaubte Spezifikationswerte: $1 \leq n \leq 6$
(natürl. Zahl)

Mit diesem Befehl wird eingestellt, mit wievielen Ziffern dezimale Satznummern ≤ 999999, evtl. unter Fortlassung führender Nullen, dargestellt werden sollen.

Zu Beginn wird dieser Wert implizit so gewählt, daß die höchste Satznummer der Datei ohne führende Nullen ausgegeben wird - ebenfalls bei einem Wechsel der Datei (siehe DATEI-Befehl).

Soll ein Satz protokolliert werden, dessen Satznummer zu groß ist, um mit diesen n Ziffern ausgegeben werden zu können, so wird der für SNRLNG eingestellte Wert implizit erhöht.

Auf Satznummern > 999999 hat diese Einstellung keinen Einfluß, da diese immer sedezimal in der Form H <Tetrade>^{12} ausgegeben werden.

10.) WANDEL

Globale Steuerung einer Umkodierung

1 Spezifikation: MODUS

erlaubte Spezifikationswerte	Wirkung
"undefiniert"	Es wird keine Umkodierung vorgenommen, alle Zeichen werden so protokolliert, wie sie in der Datei standen bzw. eingegeben wurden.
-STD-	Es findet eine Umkodierung der zu protokollierenden Sätze statt: a) Standardmäßige Umkodierung: auf dem jeweiligen Terminal nicht darstellbare Zeichen werden in die Ersatzdarstellung <Fluchtsymbol><Dezimalziffer>^{3} und das eingestellte MAL (siehe Befehl MAL) in das Fluchtsymbol mit ZC-Wert 53 umkodiert. b) Mit Hilfe des Befehles CODTAB erzeugte Umkodierungen sind wirksam.

Diese Möglichkeit des Umkodierens ist besonders wichtig bei Korrekturen im Sichtfenster-Modus (z.B. Modus SIVSAP bzw. VISTAR), da die nichtdarstellbaren Zeichen sonst als Ausrufezeichen ausgegeben und auch als solche wieder eingelesen würden.

Aber auch sonst hat sich diese Möglichkeit als sehr nützlich erwiesen - aus diesem Grund ist "WANDEL, -STD-" voreingestellt, obwohl das etwas Zeit kostet.

Die Umkodierung wirkt auf alle Befehle, bei denen Sätze protokolliert werden, außer auf die rein fernschreiberanaloge Korrektur (siehe Befehle K bzw. FSRK); da hierbei durch die Umkodierung die Möglichkeit der spaltengerechten Schreibweise verlorenginge, werden nichtdarstellbare Zeichen alle in Ausrufezeichen umkodiert - alle, d. h. auch alle Zeichen, deren Zentralcodewert kleiner als 64 ist (z. B. Steuerzeichen).

11.) CODTAB

Veränderung der standardmäßigen Umkodiertabelle

1 Spezifikation: ZEICHEN
Teilwerte zugelassen

erlaubte Spezifikationswerte:

<Spezialzeichen1>:<Spezialzeichen2>

Dabei gilt folgendes:
<Spezialzeichen1> und <Spezialzeichen2> sind beides <Spezialzeichen>, bestehen also entweder aus einem Zeichen oder aus drei den Zentralcodewert darstellenden Dezimalziffern.

Ist <Spezialzeichen1> = <Spezielzeichen2>, so wird dieses Zeichen nicht umkodiert, d.h. eine bestehende Umkodierung wird aufgehoben.

Hat <Spezialzeichen2> den Zentralcodewert 0 (Darstellung z.B. 000), so wird <Spezialzeichen1> in die Ersatzdarstellung <Fluchtsymbol><Dezimalziffer>^{3} umkodiert.

Ansonsten wird <Spezialzeichen1> in <Spezialzeichen2> umkodiert.

Diese Umkodierungen wirken natürlich nur, wenn sie eingeschaltet sind (siehe auch Befehl WANDEL).

G) SPEZIALBEFEHLE

1.) R

Raster ausgeben

1. Spezifikation: RASTERLAENGE

erlaubte Spezifikationswerte	Wirkung
"undefiniert"	Die Länge des Rasters wird gleich der aktuellen Zeilenbreite
$1 \leq n \leq$ maximale Satzlänge (n)	Es wird ein n Zeichen langes Raster ausgegeben.

2. Spezifikation: FREILASSEN

erlaubte Spezifikationswerte	Wirkung
"undefiniert"	das Raster beginnt in der ersten Spalte der nächsten Zeile.
$1 \leq n \leq$ maximale Satzlänge - Rasterlänge (n)	vor dem Raster werden n Stellen freigelassen durch ":", das Raster beginnt also bei der (n+1)-ten Spalte.

Die Ausgabe eines variablen Rasters soll ein Zählen von Spalten etc. vereinfachen.

Im allgemeinen wird man unter FREILASSEN die (aktuelle Länge der Satznummer) +1 angeben, wenn man bei der ersten Spalte eines protokollierten Satzes mit der Zählung beginnen will (siehe auch Befehl SNRLNG).

Das Raster hat folgenden Aufbau:

Jeder zehnte Punkt wird durch forlaufend zählende Ziffern dargestellt; die dazwischenliegenden Punkte (die "Fünfer") werden durch ein "+" und die übrigen Punkte durch einen "." dargestellt.

2.) STOP

Mit Hilfe dieses Befehles wird der AUFBEREITE-Lauf beendet - genauso wie durch eine leere Eingabe im Befehlsmodus.

Er erspart nur evtl. einen Konsolzyklus.

3.) SUCH

Suchen eines bestimmten Satzes und Setzen des internen Zeigers auf diesen (*A)

1., 2. und 3. Spezifikation: <Zugriff>

Wirkung:

Der interne Zeiger, der als <Satznummer> bei allen Befehlen mit *A angesprochen werden kann, wird hierbei auf den letzten laut Suchbedingung in <Bereich> gefundenen Satz gesetzt.

Typischer Anwendungsfall: "Protokolliere die ersten 20 Zeilen der Algol-Prozedur P".

Realisierung: SUCH,1+1,,/'PROCEDURE'P
P,*A+20

Dadurch, daß der interne Zeiger anschließend immer auf den letzten gefundenen Satz zeigt, sind auch bequem Anwendungen möglich in der Form:

"Protokolliere 5 Sätze vom 23. Auftreten der Zeichenfolge MAX an":
SUCH,1+23,,/MAX
P,*A+5

Man beachte jedoch, daß durch jede Protokollierung eines Satzes der interne Zeiger wieder verändert wird (siehe Einführung).

4.) SYSI

Steuerung der Systemsicherungsdatei

1. Spezifikation: MODUS

erlaubte Spezifikationswerte	Wirkung
"undefiniert"	Abschalten der Systemsicherung. Dieser Zustand ist voreingestellt und wird bei jedem Wechsel der Datei implizit eingestellt (siehe Befehl DATEI).
-STD-	Einschalten der Systemsicherung. Bei dieser Einstellung wird jedesmal, wenn ein Satz der aktuellen Datei gelöscht oder überschrieben oder einer hinzugefügt wird, der ursprüngliche Zustand des Satzes in die Datei &ZWISCHERUNG(9999.99) geschrieben, die implizit kreiert wird, falls nicht vorhanden. Dadurch ist es möglich, Änderungen, bei denen ein Fehler oder ein unerwünschter Effekt auftrat, rückgängig machen zu können. Wird ein Satz mehrmals verändert, so steht immer der Stand vor der ersten Änderung in der Sicherungsdatei.
GET	Rückgängigmachen aller in der Systemsicherungsdatei vermerkten Änderungen und Normieren der Datei (Löschen aller Sätze der Datei &ZWISCHERUNG).
PGET	dto. mit Protokoll } unter Berücksichtigung der globalen Protokolleinstellung
NGET	dto. ohne Protokoll }
INF	Informieren über die Anzahl der vermerkten Änderungen.
NORM	Normieren der Systemsicherung (Löschen aller vermerkten Änderungen).
PROT	Protokollieren aller vermerkten Änderungen.

2. Spezifikation: ZUSATZ

Diese Spezifikation hat nur bei zwei der Modi eine Bedeutung:

a) Bei MODUS = -STD- kann unter ZUSATZ der Name einer Datei angegeben werden, die als "Sicherungsdatei" benutzt werden soll, also z. B. eine LF- oder WSP Datei.

b) Bei MODUS = GET können unter Zusatz (durch Apostrophe getrennt) Satzbereiche angegeben werden, die wieder auf den Originalzustand gebracht werden sollen.

5.) AEND

Steuerung der Aenderungsdatei

1 Spezifikation: MODUS

erlaubte Spezifikationswerte: identisch wie bei Befehl SYSI.

Wirkung:

Es wird die Datei &ZWAENDERUNG(9999.99) benutzt, um alle Änderungen der aktuellen Datei zu vermerken, jedoch nicht wie beim Befehl SYSI vor der Änderung, sondern den Zustand nach der Änderung, um später alle Änderungen leicht bei einer anderen Datei wiederholen zu können.

Anwendungsbeispiel:

Man arbeite mit einer Kopie der eigentlichen LFD- oder WSP-Datei in der &STDDB, und erst wenn alle Änderungen okay sind, stelle man mit dem DATEI-Befehl auf die eigentliche Datei ein und führe mittels AEND, GET die Änderungen auch für die richtige Datei durch.

SYSI und AEND dürfen nicht beide eingeschaltet sein.

Auch hierbei wird die Änderungssicherung beim DATEI-Befehl automatisch ausgeschaltet.

6.) TUE

Dieser Befehl erlaubt es, Befehle auszuführen, die in einer Datei stehen. Er kann voll rekursiv benutzt werden.

1. Spezifikation: DATEI =[<dbn>.]<dtn>
Die Befehle stehen in der Datei <dtn> in Datenbasis <dbn>.

2. Spezifikation: BEREICH = <Bereich>

Ist DATEI = "undefiniert", so wird die aktuell bearbeitete Datei genommen.

Die Vorgehensweise ist folgende:

Alle gemäß <Bereich> aus der Datei gelesenen Sätze werden, durch das eingestellte <Zeilenende> (siehe Befehl ZENDE) getrennt, verkettet und als String anstelle des TUE-Befehles eingesetzt und sodann abgearbeitet.

7.) TIME

Mit Hilfe dieses Befehles kann die bisher im Operatorlauf verbrauchte Rechenzeit ermittelt werden.

Ausgedruckt wird immer die seit dem letzten TIME-Befehl verbrauchte Rechenzeit, bzw. beim erstenmal die seit Programmbeginn verbrauchte Rechenzeit (siehe auch Spez.-Wert TIME im PROT-Befehl).

8.) KONV

Konvertierung von Satznummern in verschiedene Darstellung

1 Spezifikation: WERT

erlaubte Spezifikationswerte:

a) negative oder positive ganze Zahl n: $n < 2^{46}$

Syntax: $[-]\langle Ziffer\rangle\left[\langle Ziffer\rangle\right]_0^{13}$

b) sedezimale Zahl .

Syntax: $H\langle Tetrade\rangle\left[\langle Tetrade\rangle\right]_0^{11}$

c) Oktaden-Satzmarke

Syntax: $\left[/\right]!\langle Oktade\rangle\left[\langle Oktade\rangle\right]_0^{5}!$

Die Fremdstringform ist zu wählen, wenn Zeichen in der Satzmarke vorkommen, die im Normalstring nicht erlaubt sind, z. B.: Apostroph, Komma, Blank etc.

Wirkung:

Der angegebene WERT wird in folgenden Darstellungen ausgegeben - unabhängig von der Darstellung der Eingabe:

a) als Satzmarke in Oktadenform
b) als Satznummer in Tetradendarstellung
c) als ganze Zahl (diese Darstellung unterbleibt, wenn es sich bei der Interpretation als Zahl um eine übergelaufene Zahl handelt).

Als zusätzliche Möglichkeit kann man als WERT auch eine gültige Gleitkommazahl im Sinne von ALGOL60 angeben, wenn man die interne Tetradendarstellung dieser Gleitkommazahl wissen möchte. Diese Zahl muß natürlich wegen der Eindeutigkeit in einerForm angegeben werden, die keine andere Darstellung sein kann. Z. B. muß 3475, wenn sie als Gleitkommazahl interpretiert werden soll, als 3475.0 angegeben werden.

VII ALPHABETISCHE LISTE ALLER BEFEHLE MIT IHREN SPEZIFIKATIONEN

Die erste, lange Liste wird von AUFBEREITE ausgedruckt beim Befehl

B,BEF,SPEZ

und die zweite, komprimierte beim Befehl

B,BEF,SPEZ'PAGE .

AEND

1 MODUS
2 ZUSATZ

B

1 NAME
2 MODUS

BER

1 BEREICH

CODTAB

1 ZEICHEN

D

1 NUMERIERUNG
2 BEREICH
3 SPALTENBEREICH
4 SUCHSTRING

DATEI

1 DATEI

DEF

1 INAM
2 SPALTENBEREICH
3 ERSETZUNGSSTRING

E

1 BEREICH
2 SPALTENBEREICH
3 SUCHSTRING

EIN

1 INFORMATION

ERRZEI

1 ERRORZEICHEN

ERS

1 AKTIV
2 MODUS
3 ANZAHL

ERSZEI

1 ERSETZUNGSZEICHEN

FSRK

1 BEREICH
2 SPALTENBEREICH
3 SUCHSTRING

GET

1 DATEI
2 NUMERIERUNG
3 BEREICH
4 SPALTENBEREICH
5 SUCHSTRING

INF

1 MODUS

K

1 BEREICH
2 SPALTENBEREICH
3 SUCHSTRING

KOMMDO

KONV

1 WERT

KORBER

1 KORREKTURBEREICH

KPROT

1 MAXLAENGE

KRIT

1 MODUS
2 GLOBALKRITERIUM

L

1 BEREICH
2 SPALTENBEREICH
3 SUCHSTRING

LOEZEI

1 LOESCHZEICHEN

MAL

1 MAL

MAXDEF

1 DEFINITIONEN

MODUS

1 MODUS

NO

1 NUMERIERUNG
2 BEREICH
3 SPALTENBEREICH
4 SUCHSTRING

NE

1 BEREICH
2 SPALTENBEREICH
3 SUCHSTRING

NEIN

1 INFORMATION

NERS

1 AKTIV
2 MODUS
3 ANZAHL

NGET

1 DATEI
2 NUMERIERUNG
3 BEREICH
4 SPALTENBEREICH
5 SUCHSTRING

NK

1 BEREICH
2 SPALTENBEREICH
3 SUCHSTRING

NL

1 BEREICH
2 SPALTENBEREICH
3 SUCHSTRING

NNUM

1 NUMERIERUNG
2 BEREICH
3 SPALTENBEREICH
4 SUCHSTRING

NS

1 NUMERIERUNG
2 INFORMATION

NUM

1 NUMERIERUNG
2 BEREICH
3 SPALTENBEREICH
4 SUCHSTRING

OP

1 BEREICH
2 SPALTENBEREICH
3 SUCHSTRING

P

1 BEREICH
2 SPALTENBEREICH
3 SUCHSTRING

PD

1 NUMERIERUNG
2 BEREICH
3 SPALTENBEREICH
4 SUCHSTRING

PE

1 BEREICH
2 SPALTENBEREICH
3 SUCHSTRING

PEIN

1 INFORMATION

PERS

1 AKTIV
2 MODUS
3 ANZAHL

PGET

1 DATEI
2 NUMERIERUNG
3 BEREICH
4 SPALTENBEREICH
5 SUCHSTRING

PK

1 BEREICH
2 SPALTENBEREICH
3 SUCHSTRING

PL

1 BEREICH
2 SPALTENBEREICH
3 SUCHSTRING

PNUM

1 NUMERIERUNG
2 BEREICH
3 SPALTENBEREICH
4 SUCHSTRING

PROT

1 PROTOKOLL

PS

1 NUMERIERUNG
2 INFORMATION

R

1 RASTERLAENGE
2 FREILASSEN

RP

1 SATZNUMMER
2 SPALTENBEREICH
3 SUCHSTRING

S

1 NUMERIERUNG
2 INFORMATION

SATZLG

1 SATZLAENGE

SNRLNG

1 SNRLAENGE

STOP

SUCH

1 BEREICH
2 SPALTENBEREICH
3 SUCHSTRING

SYSI

1 MODUS
2 ZUSATZ

TAB

1 POSITIONEN
2 TABULATORZEICHEN

TIME

TRIM

1 MODUS

TUE

1 DATEI
2 BEREICH

WANDEL

1 MODUS

ZEILBR

1 ZEILENBREITE

ZENDE

1 ZENDE

AUFBEREITE - BEFEHLE

D

MODUS
ZUSATZ

NAME
MODUS

BEREICH

TAB

ZEICHEN

NUMERIERUNG
BEREICH
SPALTENBEREICH
SUCHSTRING

EI

DATEI

INAM
SPALTENBEREICH
ERSETZUNGSSTRING

BEREICH
SPALTENBEREICH
SUCHSTRING

EIN

1 INFORMATION

ERRZEI

1 ERRORZEICHEN

ERS

1 AKTIV
2 MODUS
3 ANZAHL

ERSZEI

1 ERSETZUNGSZEICHEN

FSRK

1 BEREICH
2 SPALTENBEREICH
3 SUCHSTRING

GET

1 DATEI
2 NUMERIERUNG
3 BEREICH
4 SPALTENBEREICH
5 SUCHSTRING

INF

1 MODUS

K

1 BEREICH
2 SPALTENBEREICH
3 SUCHSTRING

KOMMOD

KONV

1 WERT

KORBER

1 KORREKTURBEREICH

KPROT

1 MAXLAENGE

KRIT

1 MODUS
2 GLOBALKRITERIUM

L

1 BEREICH
2 SPALTENBEREICH
3 SUCHSTRING

LOEZEI

1 LOESCHZEICHEN

MAL

1 MAL

MAXDEF

1 DEFINITIONEN

MODUS

1 MODUS

ND

1 NUMERIERUNG
2 BEREICH
3 SPALTENBEREICH
4 SUCHSTRING

NE

1 BEREICH
2 SPALTENBEREICH
3 SUCHSTRING

NEIN

1 INFORMATION

NERS

1 AKTIV
2 MODUS
3 ANZAHL

NGET

1 DATEI
2 NUMERIERUNG
3 BEREICH
4 SPALTENBEREICH
5 SUCHSTRING

NK

1 BEREICH
2 SPALTENBEREICH
3 SUCHSTRING

NL

1 BEREICH
2 SPALTENBEREICH
3 SUCHSTRING

A U F B E R E I T E - B E F E H L E

NUM

1 NUMERIERUNG
2 BEREICH
3 SPALTENBEREICH
4 SUCHSTRING

S

1 NUMERIERUNG
2 INFORMATION

UM

1 NUMERIERUNG
2 BEREICH
3 SPALTENBEREICH
4 SUCHSTRING

P

1 BEREICH
2 SPALTENBEREICH
3 SUCHSTRING

1 BEREICH
2 SPALTENBEREICH
3 SUCHSTRING

D

1 NUMERIERUNG
2 BEREICH
3 SPALTENBEREICH
4 SUCHSTRING

E

1 BEREICH
2 SPALTENBEREICH
3 SUCHSTRING

PEIN

1 INFORMATION

PERS

1 AKTIV
2 MODUS
3 ANZAHL

PGET

1 DATEI
2 NUMERIERUNG
3 BEREICH
4 SPALTENBEREICH
5 SUCHSTRING

PK

1 BEREICH
2 SPALTENBEREICH
3 SUCHSTRING

PL

1 BEREICH
2 SPALTENBEREICH
3 SUCHSTRING

PNUM

1 NUMERIERUNG
2 BEREICH
3 SPALTENBEREICH
4 SUCHSTRING

PROT

1 PROTOKOLL

PS

1 NUMERIERUNG
2 INFORMATION

R

1 RASTERLAENGE
2 FREILASSEN

RP

1 SATZNUMMER
2 SPALTENBEREICH
3 SUCHSTRING

S

1 NUMERIERUNG
2 INFORMATION

SATZLG

1 SATZLAENGE

SNRLNG

1 SNRLAENGE

STOP

SUCH

1 BEREICH
2 SPALTENBEREICH
3 SUCHSTRING

SYSI

1 MODUS
2 ZUSATZ

TAB

1 POSITIONEN
2 TABULATORZEICHEN

TIME

TRIM

1 MODUS

TUE

1 DATEI
2 BEREICH

WANDEL

1 MODUS

ZEILBR

1 ZEILENBREITE

ZENDE

1 ZENDE

VIII LITERATURVERZEICHNIS

1 R. Buchmann
BODAT, ein schnelles und platzsparendes System zur Datenmanipulation und -speicherung in ALGOL60 und FORTRAN (2. ergänzte Auflage)
Arbeitsberichte des Rechenzentrums der Ruhr-Universität Bochum, Nr. 7405 (1974)

2 M. Rosendahl
BOGOL-STRING, eine flexible Zeichenkettenverarbeitung in ALGOL60
(2. erweiterte und geänderte Version)
Arbeitsberichte des Rechenzentrums der Ruhr-Universität Bochum, Nr. 7602 (1976)

3 R. Buchmann u. M. Rosendahl
BOGOL-TAS, eine Spracherweiterung von ALGOL60 durch Codeprozeduren zur Systemprogrammierung
Arbeitsberichte des Rechenzentrums der Ruhr-Universität Bochum, Nr. 7603 (1976)

4 R. Buchmann
RB&QUELLHALT, ein TR440-Datenbanksystem zur platzsparenden Quellhaltung auf Datenträgern mit direktem Zugriff (LFD, WSP)
Arbeitsberichte des Rechenzentrum der Ruhr-Universität Bochum, Nr. 7305 (1973

5 KOMMANDOSPRACHE TR 440

Bisher erschienene Arbeitsberichte des Rechenzentrums

der Ruhr-Universität Bochum

Nr. 7101: K.-H. Mohn, M. Rosendahl, H. Zoller
AIDA; eine Dialogsprache für den TR 440 (vergriffen)

Nr. 7102: K.-H. Mohn, M. Rosendahl, H. Zoller
AIDA, ein Dialogsystem und seine Implementierung in ALGOL (vergriffen)

Nr. 7103: K.-H. Mohn, M. Rosendahl, H. Zoller
AIDA, Manual für den Benutzer (vergriffen)

Nr. 7104: 4. Jahresbericht des Rechenzentrums (Juni 1970 bis Juni 1971)

Nr. 7105: H. Wupper
WR 1302 - Ein einfaches Band-Betriebssystem für einen mittleren Rechner

Nr. 7201: H. Windauer
Existenzsätze zur (0,1,..,R-2,R) - Interpolation

Nr. 7202: W. Schelongowski
DIATRACE - Ein System zur interaktiven Assemblerprogrammierung

Nr. 7203: M. Jäger, M. Rosendahl, R. Staake
Einführung in die Listenverarbeitung anhand der Dialogsprache AIDA

Nr. 7204: R. Mannshardt, P. Pottinger
Einführung in die Benutzung des Teilnehmer-Rechensystems TR 440 in der RUB (vergriffen)

Nr. 7205: 5. Jahresbericht des Rechenzentrums (1.7.1971 bis 30.6.1972)

Nr. 7206: M. Rosendahl
BOGOL-TAS, ein Weg zur systemnahen Programmierung in ALGOL am TR 440

Nr. 7207: W. Stark
ILW, Programmsystem zur Berechnung des Instationären Ladungswechsels von Verbrennungskraftmaschinen (Modulbeschreibung und Eingabekonventionen)

Nr. 7208: W. Stark
ILW, Programmsystem zur Berechnung des Instationären Ladungswechsels von Verbrennungskraftmaschinen (Regelmechanismus und Berechnung der Rohrströmung)

Nr. 7209: H. Ehlich
Anregung und Kritik zum Betriebs- und Programmiersystem der TR 440

Nr. 7210: M. Rosendahl
BOGOL-STRING, eine flexible Zeichenkettenverarbeitung in ALGOL 60

Nr. 7211: H. Camici, H. Claus, H. Ehlich, D. Kipp
Arbeitsbericht über ein Programm zur Haushaltsführung

Nr. 7301: R. Mannshardt, K.-H. Mohn, H. Münch, P. Pottinger
Einführung in die Benutzung des Teilnehmer Rechensystems TR 440
2. geänderte Auflage (vergriffen)

Nr. 7302: K.-H. Mohn
Über einige Anwendungen des Computers in der Medizin

Nr. 7303: R. Buchmann
BODAT, ein schnelles und platzsparendes System zur Datenmanipulation und -speicherung in ALGOL 60 und FORTRAN

Nr. 7304: M. Hauenschild
Ansätze zur komplexen Kreisarithmetik

Nr. 7305: R. Buchmannn
RB&QUELLHALT, ein TR440-Datenbanksystem zur platzsparenden Quellhaltung auf Datenträgern mit direktrem Zugriff (LFD, WSP)

Nr. 7306: 6. Jahresbericht des Rechenzentrums (1.7.1972 bis 31.12.1973)

Nr. 7401: R. Buchmann
Der Systemoperator BO&BS3OP
Messungen und Steuerungen des Betriebssystems auf Operatorebene

Nr. 7402: R. Mannshardt
Herleitung und Prüfung spezieller Runge-Kutta-Verfahren mit einem impliziten Rechenschritt

Nr. 7403: R. Buchmann, H. Wupper
Unzulänglichkeiten des TR 440 Programmiersystems und ihre Umgehung

Nr. 7404: R. Green, K.-H. Mohn
Quellbezogene FORTRAN Optimierungen für den Compiler des TR 440

Nr. 7405: R. Buchmann
BODAT, ein schnelles und platzsparendes System zur Datenmanipulation und -speicherung in ALGOL 60 und FORTRAN (2..ergänzte Auflage)

Nr. 7501: R. Buchmann
Zur Theorie der Montage von Programmoduln

Nr. 7502: 7. Jahresbericht des Rechenzentrums (1.1. bis 31.12.1974)

Nr. 7503: H.-D. Sander
BOTRAN, eine Fortran Spracherweiterung durch Code-Prozeduren

Nr. 7504: W. Schelongowski
DIATRACE - Ein System zur interaktiven Assemblerprogrammierung

Nr. 7505: Camici, Prof. Dr. Ehlich, Schürmann
Über ein Programm zur Material- und Vervielfältigungs-Abrechnung

Nr. 7506: Camici, Prof. Dr. Ehlich, Herrmannies
Über ein Programm für die Telefonabrechnung einer Nebenstellenanlage

Nr. 7507: Camici, Prof. Dr. Ehlich, Kipp
Über ein Programm zur Haushaltsführung

Nr. 7508: Camici, Cipa, Prof. Dr. Ehlich
Bericht über ein Programm zu Verwaltung der Studentendaten

Nr. 7509: J. Riege
Zur mehrdimensionalen Spline-Interpolation bezüglich beliebiger linearer Funktionale

Nr. 7601: H. Ehlich, J. Riege u. K.-H. Schloßer
Ein Programmsystem zur Ausleihverbuchung und interaktiven Rechnerunterstützung in der allgemeinen Buchbestandsverwaltung
Teil 1: Offline-System

Nr. 7602: M. Rosendahl
BOGOL-STRING, eine flexible Zeichenkettenverarbeitung in ALGOL60
2. erweiterte und geänderte Version

Nr. 7603: R. Buchmann, M. Rosendahl
BOGOL-TAS, eine Spracherweiterung von ALGOL 60 durch Codeprozeduren zur Systemprogrammierung

Nr. 7604: R. Buchmann
AUFBEREITE, ein universell einsetzbarer Dateiänderungsoperator für verschiedene Datei- und Dialoggerättypen und/oder Betriebsarten